2012-06

UAV to UAV Target Detection and Pose Estimation

Hajri, Riadh

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

UAV TO UAV TARGET DETECTION AND POSE ESTIMATION

by

Riadh Hajri

June 2012

Thesis Advisor:	Timothy H. Chung
Second Reader:	Raymond Buettner

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704–0188

1. REPORT DATE *(DD–MM–YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From — To)*
6–6–2012	Master's Thesis	21-06-2010 – 15-06-2012

4. TITLE AND SUBTITLE

UAV to UAV Target Detection and Pose Estimation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Riadh Hajri

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

13. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.IRB Protocol Number: NA.

14. ABSTRACT

The objective of this thesis is to investigate the feasibility of using computer vision to provide robust sensing capabilities suitable for the purpose of UAV to UAV detection and pose estimation using affordable CCD cameras and open coding libraries. We accomplish this by reviewing past literature about UAV detection and pose estimation and exploring comparison of multiple state-of-the-art algorithms. The thesis presents implementation studies of detection approaches including color-based detection and component-based detection. We also present studies of pose estimation methods including the PosIt algorithm, homography-based detection, and the EPFL non-iterative method. The thesis provides a preliminary strategy for detecting small UAVs and for estimating its six degree of freedom (6DOF) pose from image sequences within the prescribed airspace. Discussion of its performance in processing synthetic data is highlighted for future applications using real-life data sets.

15. SUBJECT TERMS

UAV detection, Pose estimation, Computer Vision, Obstacle Avoidance, Edge Detection, Morphological Filtering.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			19b. TELEPHONE NUMBER *(include area code)*
Unclassified	Unclassified	Unclassified	UU	89	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

UAV TO UAV TARGET DETECTION AND POSE ESTIMATION

Riadh Hajri
Captain,Tunisian Air Force
Systems and Network Engineer, Tunisian Air Force Academy, June 2003

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS TECHNOLOGY

from the

NAVAL POSTGRADUATE SCHOOL
June 2012

Author: Riadh Hajri

Approved by: Timothy H. Chung
Thesis Advisor

Raymond Buettner
Second Reader

Dan Boger
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The objective of this thesis is to investigate the feasibility of using computer vision to provide robust sensing capabilities suitable for the purpose of UAV to UAV detection and pose estimation using affordable CCD cameras and open coding libraries. We accomplish this by reviewing past literature about UAV detection and pose estimation and exploring comparison of multiple state-of-the-art algorithms. The thesis presents implementation studies of detection approaches including color-based detection and component-based detection. We also present studies of pose estimation methods including the PosIt algorithm, homography-based detection, and the EPFL non-iterative method. The thesis provides a preliminary strategy for detecting small UAVs and for estimating its six degree of freedom (6DOF) pose from image sequences within the prescribed airspace. Discussion of its performance in processing synthetic data is highlighted for future applications using real-life data sets.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

List of Figures

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

UAV	Unmanned Aerial Vehicle
UAS	Unmanned Aerial System
CAS	Collision Avoidance System
NAS	National Airspace
FAA	Federal Administration Agency
TCAS	Traffic Alert and Collision Avoidance System
DoD	Department of Defense
6DOF	six Degrees of Freedom
IMU	Inertial Measurement Unit
DGPS	Differential Global Positioning System
FOV	Field Of View
AGL	Above Ground Level
MAC	Mid-Air Collision
VO	Vision Odometry
VFR	Visual Flight Rules
FPS	Frames Per Second
RGB	Red, Green, Blue
ROI	Region of Interest
SOP	Scaled Orthographic Projection

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

The objective of this thesis is to investigate the feasibility of using computer vision to provide robust sensing capabilities suitable for the purpose of UAV to UAV detection and pose estimation using affordable CCD cameras and open coding libraries. We accomplish this by reviewing past literature about UAV detection and pose estimation and exploring comparison of multiple state-of-the-art algorithms. The thesis presents implementation studies of detection approaches including color-based detection and component-based detection. We also present studies of pose estimation methods including the PosIt algorithm, homography-based detection, and the EPFL non-iterative method. The thesis provides a preliminary strategy for detecting small UAVs and for estimating its six degree of freedom (6DOF) pose from image sequences within the prescribed airspace. Discussion of its performance in processing synthetic data is highlighted for future applications using real-life data sets.

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgements

All praise to Allah Most Gracious, Most Merciful, Who, Alone, brings forgiveness and light and new life to those who call upon Him; and to Him is the dedication of this work.

I gratefully acknowledge my advisor Professor Timothy Chung for his patient guidance and thoughtful insight throughout this effort. I would also like to thank my second reader, Professor Raymond Buettner, for his time and patience.

To my loving and supportive wife, Khira, who has always been dedicated to our family and my career. I could not have done this without you.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction

1.1 Motivation

The collision of the Cessna 172 in Visual Flight Rules (VFR) conditions with a Cirrus SR22 in Aug 10, 2008 put to the test "big sky, little plane" theory, that states that wide-open spaces generally prevent two aircraft from colliding. The National Transportation Safety Board determined that the probable cause(s) of this accident was the failure of both pilots to see and avoid each other's aircraft [1]. Table 1.1 gives statistics on the number of Mid-Air Collision (MAC) in U.S. airspace between 1991 and 2002.

Year	MAC Events	Operating Hours (millions)	Rate per 10^6 Flight Hours
1991	18	27.2	0.66
1992	11	24.8	0.44
1993	13	22.8	0.57
1994	11	22.2	0.50
1995	14	24.9	0.56
1996	18	24.9	0.72
1997	13	25.5	0.51
1998	14	26.8	0.52
1999	15	29.5	0.51
2000	19	30.8	0.62
2001	5	25.9	0.19
2002	7	25.9	0.27
Average	13.17	25.93	0.51

Table 1.1: This table shows the number of MAC for each year starting from 1991 to 2002 for the number of operation hours. We can see that the year 1991 has the highest MAC rate and 2001 has the lowest rate (from: [2]).

The introduction of Unmanned Aerial Vehicle (UAV)s into the National Airspace (NAS) is challenging for the Federal Administration Agency (FAA) and the aviation community. Under existing regulations in the U.S., UAV operations require special approval if operating higher than 500 feet Above Ground Level (AGL) or over populated areas [3]. Furthermore, UAV operations beyond Line-of-Sight (LOS) may require an automated sense-and-avoid (S& A) system due to potential communications latencies or failures. FAA policy to date has been to enforce highly restrictive operating requirements that involve segregating UAV operations from those

of piloted aircraft, and confining the former to specially designated airspaces. Table 1.2 depicts the azimuth and elevation boundaries.

Source	Azimuth	Elevation
FAA P-8740-51: How to Avoid a Mid-Air Collision	+/- 60 degrees	+/- 10 degrees
International Standards, Rules of the Air, Section 3.2 (ICAO)	+/- 110 degrees	No guidance
FAA Advisory Circular 25.773-1 (Transport Aircraft Design) degrees (varies with azimuth)	+/- 120 degrees	Variable: +37 and -25

Table 1.2: Existing Guidance for Detection Search Areas (from: [2])

In addition to the safety concern, a major threat is that future warfighters may have to face a UAS-equipped opponent [4] since many players have been active in acquiring their own UAVs, requiring allied nations to refine techniques to counter hostile drones. In this respect Department of Defense (DoD) has begun a series of research projects and exercises to test out what functionality should the UAV offer to detect and counter such a threat. Thus the research detailed in this thesis is part of an ongoing research project on "Swarm vs. Swarm Unmanned Systems." In this thesis UAV to UAV detection is considered one of the most important issues to be addressed for both collision avoidance and any manoeuvre against enemy UAV. The DoD performs a yearly exercise known as "Black Dart," which aims to check the ability of a variety of manned and unmanned aerial systems to detect and identify enemy Unmanned Aerial System (UAS) platforms in flight. Figure 1.1 illustrates the overall components of the project and fits this thesis's research into the context of the larger project.

Whether the case is to safely fly UAVs, in the NAS or to counter enemy UAS, an automated "sense and avoid" system is key to enabling UAVs to routinely fly within shared airspace. That is, an S&A system should be able to distinguish the target UAV from the background with a performance that equals or exceeds that of a human pilot, as stated in flight regulations. Such a detection system should provide:

- High detection rates
- Low false alert rates
- Early detection
- Real-time operation

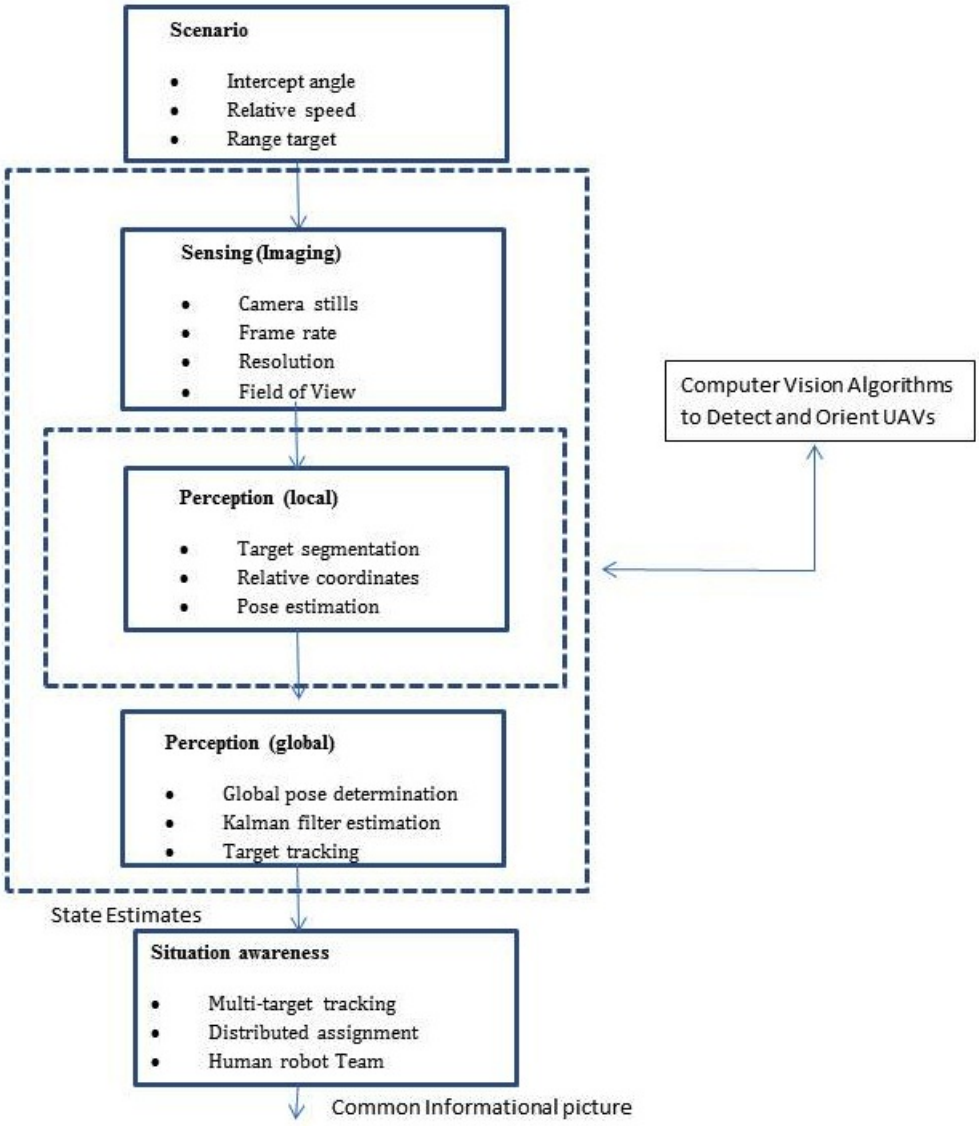


Figure 1.1: Swarm vs. Swarm Unmanned Systems: This figure shows how the whole project is organized and subdivided into smaller projects. In most cases, these smaller projects are theses being worked on by Naval Postgraduate School students.. This thesis addresses local perception, which deals with target segmentation, relative coordinates, and pose estimation using computer vision.

A general description of a sense and avoid system will involve the following steps:

- Sensing: obtaining accurate and reliable information about the airspace where the UAV operates.
- Detection: Identify and prioritize collision threats based on the information sensed.
- Action: Determining Manoeuvre necessary to maintain safe separation or to engage in

some kind of operations.

Figure 1.2 illustrates the general description of such a sense and avoid system.

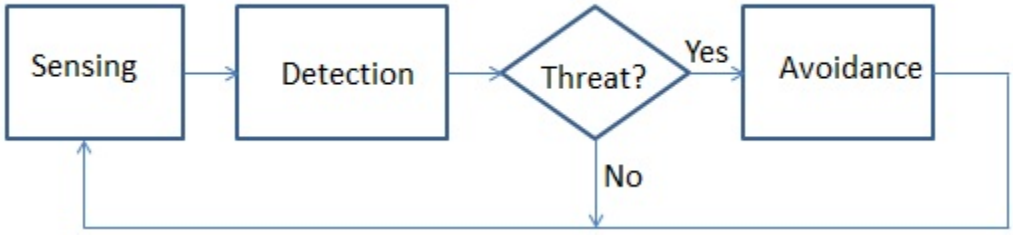


Figure 1.2: General S&A system (from: [5]): It is a Sensing Detection Threat identification and Avoidance (a.k.a. SDTA) loop where the surrounding environment is continuously sensed, potential obstacles are detected followed by a decision whether they are considered as threat or not and then the avoidance manoeuvre is performed

The detection step is considered one of the most important component of the collision avoidance system because of the critical information that it provides to systems responsible for determining and executing appropriate action [5].

1.2 Literature Review

The global objective of the Collision Avoidance System (CAS) is to allow UAVs to operate safely within the non-segregated civil and military airspace on a routine basis. Generally speaking, a CAS can be achieved via cooperative or non-cooperative means, and the physical sensors that collect the information may be categorized as either active or passive.

1.2.1 Sensors

UAV target designation devices are either passive or active sensors. Passive sensors rely on radiated energy from the targets of interest. Passive sensors include optical cameras (such as CMOS or CCD cameras), infrared cameras such as forward looking infrared (FLIR), and radio receivers as well as radio direction finding equipment. Active sensors, on the other hand, require interaction with targets. They transmit energy and detect the energy reflected directly or indirectly from these targets. There are several target acquisition sensors which can be carried by each UAV as a payload. They are:

Electro-Optics

Optical sensors include both imaging and non-imaging sensors; many are based on the use of lasers, imaging systems. Among these sensors we can mention FLIR or thermal imagers, which is a device that enhances target acquisition in low visibility situation utilizing heat or thermal images from the target. Many UAVs carry IR sensors on their payloads that detect heat as is the case of the Global Hawk UAV [6] (along with synthetic-aperture radar as well as electro-optical sensors). As a case study of the IR sensor's effectiveness, it was used to assess the damage inside the Japanese nuclear reactors after the 2011 earthquake and tsunami. Its IR sensors were able to acquire images of the reactors showing which parts of them were at what temperatures. IR requires heat from an object to perform object detection, which is most suitable for night-time use. In addition to FLIR, we can mention high resolution charge coupled device (CCD) cameras, which can help target detection and recognition during daylight where weather is clear. It requires an advanced image processing and pattern recognition algorithm to analyze picture and video imagery for target recognition tasks.

Radar/Lidar

Radar is the sensor of choice for long-range collision avoidance [2]; the problem with radar is that long-range sensors require a lot of power and beam localization requires a large antenna neither of which is suitable for small UAVs. Some work has been done to miniaturize radar so that it will fit in small UAVs, but it is still in the experimental phase. Synthetic aperture radar (SAR) and Light Detection And Ranging (Lidar) are also used as active sensors for various missions, especially for target recognition and detection. Such active sensors can be helped by the use of a moving target indicator (MTI) which enables the radar to designate a moving target and engage the target continuously until a decision is made concerning the target or it is outside the radar's detection range.

Acoustic Sensors

Acoustic wave sensors are electronic devices that can measure sound levels [7]. When an acoustic wave travels through a certain material or along the surface of a material, it is influenced by the different material properties and obstacles it travels through. Any changes to the characteristics of this path affect the velocity and/or amplitude of the wave. These characteristics are translated into a digital signal (output) using transducers and can be monitored by measuring the frequency or phase characteristics of the sensor. These changes can then be translated to the corresponding physical differences being measured. For example, Scientific Applications & Research Associates, Inc. has developed a compact acoustic sensor system for detecting aircraft

known as Passive Acoustic Non-Cooperative Collision-Alert System (PANCAS) [2]. PANCAS works by detecting the noise generated by aircraft. This system shows good performance in adverse weather and battlefield conditions, but it suffers from low bearing resolution. The data gathered from acoustic sensor can be used to cue optical sensors and other narrower Field Of View (FOV) sensors to potential targets.

1.2.2 Operational Mode

Cooperative Collision Detection

A cooperative collision detection system includes all communications equipment that enables exchange information (such as position, heading, speed and waypoints) between the cooperative aircraft. Among these devices there is the Traffic Alert and Collision Avoidance System (TCAS) [8], which is based on a transponder and provides a highly reliable detection sensor for cooperative threats. Other emerging technologies include Airborne Separation Assistance Systems (ASAS) and Automatic Dependent Surveillance Broadcast (ADS-B) [9] [10]. The Military Airborne Surveillance System (MASS) is the military counterpart to TCAS. Though designed for manned aircraft, TCAS will likely work with larger UAVs though it may present problems with smaller UAVs, which have limited payloads and low electrical generation capabilities. Another drawback of TCAS is seen when one of the UAVs fails to cooperate with others; in this case, the system becomes useless.

Non-Cooperative Collision Detection

In non-cooperative collision detection systems, the solution requires new sensors from the available technologies including laser range finders, optical flow sensors such as Electro-Optical/Infra-Red (EO/IR), radar systems or stereo camera pairs, or single moving camera. These sensors provide situation awareness about the surrounding environment and all the processing and decisions are made by the concerned aircraft without cooperation or feedback from others.

1.2.3 Related Works

This section introduces an extensive UAV target detection literature followed by works related to UAV pose estimation while focusing on computer vision methods as the selected approach.

Target Detection

There has been extensive research on the subject of target detection and, more precisely, UAV collision avoidance. Current research in this field has experimented with a variety of sensor technologies, such as radar, laser range finders, sonar sensors, infrared sensors, and cameras.

The use of long-range high-resolution laser scanners on larger UAVs provided successful navigation in cluttered environments [11]. Nevertheless, these UAVs are constrained to use most of their payload capability to carry the laser scanner. Although laser scanners have been miniaturized for use on small and Micro Unmanned Vehicles, the result is a loss of both resolution and sensing directions [12]. Radar has been also used as a reliable detection sensor suitable for large UAVs, to the author's knowledge there is no miniaturized implementation of radar to fit in small UAVs; in addition, long-range radar requires significant power and a large antenna to localize the beam.

The first feasibility study of hear-and-avoid was presented in [13] using acoustic vector sensors on small UAVs; the study's findings suggest that sounds as produced by civil aircraft will be clearly detectable by the sensors, but that will not be effective for electrically powered UAVs that produce very little sound.

An extensive focus has been given to the use of cameras for outdoor UAV sense-and-avoid. This is because the camera is a passive sensor that consumes less energy compared to radar, laser scanners, and active sensors in general and also because it has a relatively light weight and fits easily in small UAVs.

S. Grzonka *et al*. [12] used passive, stereo image-based obstacle detection. The main drawback of the use of stereo vision is that it has a limited detection range, which depends on the resolution of the images and also the distance between the two cameras. Symington *et al*. presented in [14] a probabilistic target detection by camera-equipped UAVs which considered the problem of detecting and tracking a static target from a camera mounted below a quadrotor UAV. For purposes of image registration, Speeded-Up Robust Features (SURF) were used [15].

The results of experiments with multi-source remote sensing images show that the approach used can achieve acceptable accuracy and satisfy real-time demand. However, registration error is difficult to analyse because it is hard to distinguish wrong features from features of the area with local distortion. To solve this problem, a better feature-matching method should be researched to eliminate incorrectly matched pairs [15].

Dae-Yeon and Tahk explored in [16] the use of a light source-aided computer vision system for UAV landing using some colored LED (red and green) on the edge of the recovery net and devising an algorithm to accurately detect these lights using image processing stage.

Continued advances in the fields of image-processing and computer vision have pro-

moted their suitability for target detection. That is, extensive work has been published such as in [17] and [18]. The majority of the research that has been published on the topic of automatic target detection using computer vision employs different approaches including spatial techniques, such as morphological filtering [19] which used a close-minus-open filter to extract point-like features from large-scale clutter, such as clouds. The preliminary analysis of the data set has yielded encouraging results, achieving first detection times at distances of approximately 6.5km (3.5nmi), and median filters [20] which deal with the problem of detection and tracking of low observable small-targets from a sequence of IR images against a structural background and non-stationary clutter, and temporal-based techniques such as 3D matched filtering [21] and dynamic programming [22].

Pose Estimation

Pose estimation is considered to be a main function for UAV navigation which aims to determine the position in term of x, y, and z coordinates and orientation in term of roll, pitch, and roll angles (Chapter 2 provides more details about the subject) of an object, i.e., its six Degrees of Freedom (6DOF). There are many approaches and technologies to detect and track the pose of an object. For example, mechanical, magnetic, inertial, vision, and hybrid solutions exist, each having its pros and cons [23]. Current systems use Inertial Measurement Unit (IMU)s with Differential Global Positioning System (DGPS); however, DGPS requires an active communication channel between the two aircraft.

There has been a variety of indoor pose estimation systems developed for aerial vehicles. A well-known algorithm for pose estimation is the PosIt (Pose with Iterations) algorithm, which estimates the pose of the camera with respect to an object and optimizes the error by using an iterative process. This algorithm will be one of the approaches studied and implemented in the current thesis.

Much of the prior research in using visual information for pose estimation has been in Vision Odometry (VO) where a pose is computed directly from the motion of points in the camera view. Another approach focuses on fusing information from a camera with a GPS/IMU system to achieve more accurate pose estimation. In this respect, two fusion approaches were used:

- The first uses visual information to gauge the accuracy of the GPS/IMU estimates [24], which requires that the 3D location of points in the camera view be accurately estimated.
- The second approach directly estimates pose by aligning successive images; this method

can reach accurate results but requires extensive computation time.

Computer vision (a.k.a CV) has been largely used in pose estimation for both co-planar and non-coplanar features points. In 1995, Daniel DeMenthon published a method called Pose from Orthographic Projection and Scaling with Iterations (PosIt) [25] for non-coplanar points. Its is a fast method that works with orthographic projection instead of perspective, which is simpler to work with and can approximate the correct pose. DeMenthon also developed another method [26] because the first version of PosIt does not work well for planar or close to planar point clouds. The coplanar case will be the focus of this thesis, since the shape of the UAV of interest is nearly planar.

Yang *et al*. [27] used computation of homography between planes for pose estimation based on four coplanar points. This study achieved promising results in determining the pose of four markers on a compact disk. Vincent Lepetit *et al*. developed a non-iterative pose estimation [28] method that works well for many points, since the system of equations has the same computation time for higher amount of points. This method uses four control points that are not coplanar, and it defines all points according to these control points so only the control points need to be found.

1.3 Main Contributions

This thesis shows the feasibility of UAV detection and pose estimation using commercially available off-the-shelf (COTS) camera and computer vision approaches and libraries. The results of this thesis can be used as a starting point in a future design and implementation of real-time vision based UAV detection and pose estimation. This thesis serves also as proof of concept and foundation for computer vision techniques for both UAV detection and pose estimation.

1.4 Organization of Thesis

Chapter 1 provides background and condensed literature on the various components of collision avoidance system. Chapter 2 presents conventions, coordinates frames and camera model and projections that will be used by computer vision algorithms. Chapter 3 explores two computer vision algorithms for UAV *detection*, and presents their implementation and experimental results. Chapter 4 presents three *pose estimation* methods, discusses the simulation environment, and evaluates the performance of each algorithm. Chapter 5 summarizes findings and explores follow-on research questions. The thesis organization is depicted graphically in Figure 1.3.

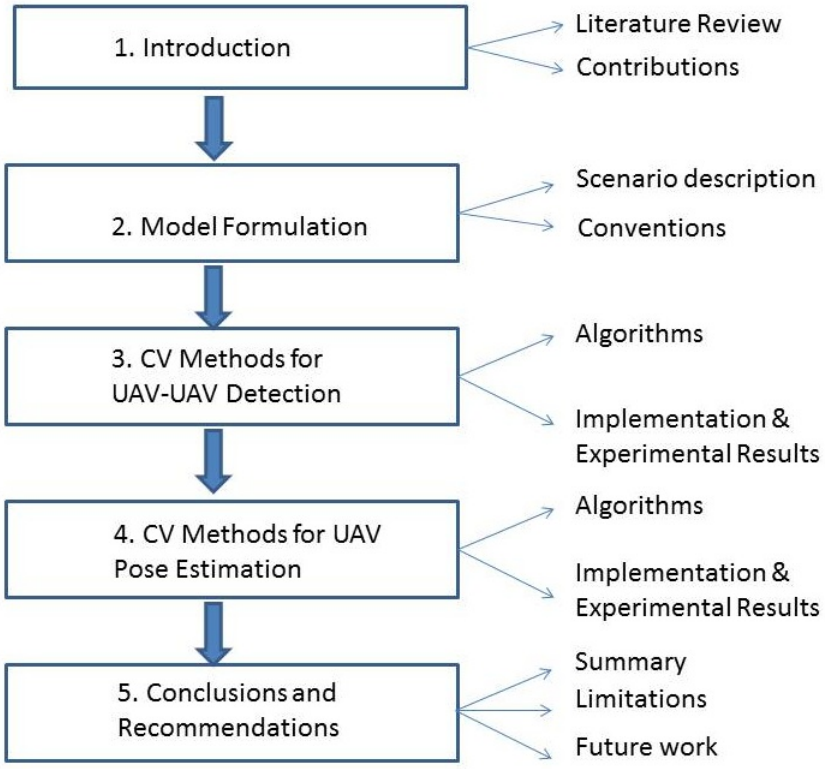


Figure 1.3: Thesis Organization: After presenting a literature review, a description of different scenarios and conventions used is provided. UAV to UAV detection methods are discussed, algorithms implementation are provided, and results are discussed. Three pose estimation approach are analyzed, implemented and tested. The last chapter discusses conclusions and recommendations.

CHAPTER 2:
Model Formulation

This chapter details the modeling process for target detection and pose estimation. Included in the chapter are descriptions of scenarios and evaluation measures, different coordinate frames, camera models, and projections that will be used in computer vision algorithms to perform both detection and pose estimation.

2.1 Scenario Development

While flying in a shared airspace the UAV should be able to detect and localize the target UAV in both collision course where target flies head on directly toward the host UAV (Figure 2.1) and crossing object manoeuvre where target UAV flies perpendicular to the host UAV (Figure 2.2). The work will be based on the following assumptions:

- Flight speed is fixed at around 50 knots
  If the velocity of the host UAV is denote by V_h, and the velocity of the target UAV) as V_t then $V_h = V_t = 50$ knots.
- Airspace dimensions are 4km by 500m by 2km
- No counter manoeuvre or chasing is envisaged

Minimum Time for Detection

In the case of collision course manoeuvre, the target UAV will not appear to be moving in the image plane because of its low relative velocity. For this reason, the detection will be difficult and the time left to both detect and perform any action will be short. This time is function of the two vehicles' range of speeds and banking angles (banking angle, or roll, is the angle at which the vehicle is inclined about its longitudinal axis with respect to its path).

In order to avert collision, the host UAV needs to begin the avoidance manoeuvre no less than $t_{\text{min\_avoid}}$ seconds, which is translated as starting the manoeuvre when the target UAV is outside the radius $r_{\text{min}} = V_{\text{clos}}\, t_{\text{min\_avoid}}$, where V_{clos} is the closing velocity (assuming exact head-on collision course, as in Figure 2.1), given by the following equation [2]:

$$V_{\text{clos}} = V_h + V_t \tag{2.1}$$

It has been shown in [2] that, for for the instantaneous banking angle $\phi_{\max}$, the minimum required time to make decision to avoid the target UAV. (It is the last chance to avoid a collision).

$$t_{\text{min\_avoid}} = \sqrt{2\, r_{\min}\, \cot(\phi_{\max})} = 5.6\, \sqrt{\pi/2 - \phi_{\max}} \tag{2.2}$$

The approximate distance between the two aircraft at the start of the collision avoidance manoeuvre must be:

$$d_{\text{avoid}} = V_{\text{clos}}\, t_{\text{min\_avoid}} = \sqrt{2\, r_{\min}\, \cot(\phi_{\max})} = 5.6 V_{\text{clos}} \sqrt{\pi/2 - \phi_{\max}} \tag{2.3}$$

These computed parameters provide constraints for the real-time computation required to detect and identify potential targets in real-time which will be measure of effectiveness of any proposed approach.

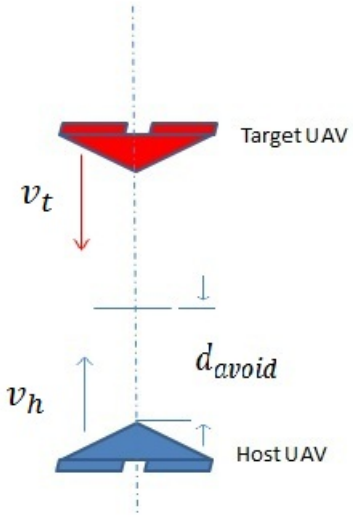


Figure 2.1: Collision Course Manoeuvre: Both host and target UAV are flying at constant speed v_t = v_h = 50 knots. They are flying head on a collision course. The distance d_{avoid} is the threshold for collision avoidance

Minimum Detection Arc Width

This section attempts to determine what the size of the target UAV must be in order to allow detection to avoid a probable collision. The angle α subtended by the target UAV of width w as viewed at a distance d [2] is given by

$$\alpha = w/d \tag{2.4}$$

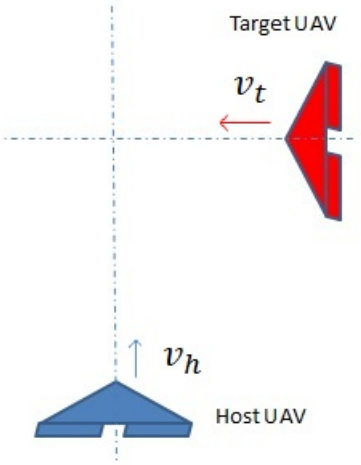


Figure 2.2: Crossing Manoeuvre:Target UAV is flying perpendicular to the flight direction of the host UAV. Both UAVs are flying at the same speed v_t = v_h =50 knots.

This case uses a fixed CCD camera so no revisit time: since it is not a scanning sensor. The total time could take up to

$$t_{\text{total}} = t_{\text{detect}} + t_{\text{min\_avoid}} \tag{2.5}$$

And the angle subtended in this situation is:

$$\alpha = w/(t_{\text{total}} * V_{\text{clos}}) \tag{2.6}$$

Therefore for a given target UAV with a velocity V_t, the sensing system needs to be able to detect it whose arc width is at least as small as α calculated above.

2.2 Conventions

2.2.1 Coordinate Systems

Multiple coordinate systems are often used to describe the positions of an object in a complex environment.

UAV Body Coordinate Frame

The UAV body frame is noted by {B} and has its coordinates axis X^B, Y^B, Z^B as shown in Figure 2.3. The rotation angles are defined as:

- **roll:** ϕ^B refers to whether the body is upside-down or not, i.e., its orientation within the

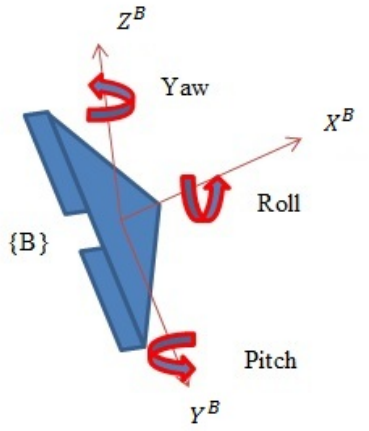


Figure 2.3: UAV Body Coordinate Frame:The frame origin is at the UAV's center of gravity, the x-axis points forward along the longitudinal axis of the aircraft, the y-axis points outwards towards the right wing, and the z-axis is in the upward direction.

Y^BZ^B plane, or rotating around the X^B axis

- **pitch:** θ^B refers to whether the body is tilted, i.e., its orientation within the X^BZ^B plane, or rotating around the Y^B axis
- **yaw:** φ^B refers to the direction in which the body is facing i.e., its orientation within the X^BY^B plane, or rotating around the Z^B axis.

Camera Coordinate Frame

This is a right-hand orthogonal coordinate system whose origin is located at the focal point of the camera, where the X^C-axis points forward along the longitudinal axis of the camera, the Y^C-axis points outwards toward the right hand side, and the Z^C-axis points downward from the origin (Figure 2.4).

Image Plane Coordinate Frame

The camera model used in this thesis is the pinhole camera. This model can be used to map a three dimensional coordinates of a series of three dimensional coordinates of a point M_i, represented in the camera coordinate frame to two dimensional coordinates in Image Plane Coordinate Frame.

The coordinates of an object point M_i in the camera coordinate frame is $M_i^C = [X_i^C, Y_i^C, Z_i^C]^T$ and its projection onto the image plane is the point m_i whose coordinate in the image plane coordinate frame is $[x_i^I, x_i^I]^T$ as shown on Figure 2.4. The equation that expresses the mapping

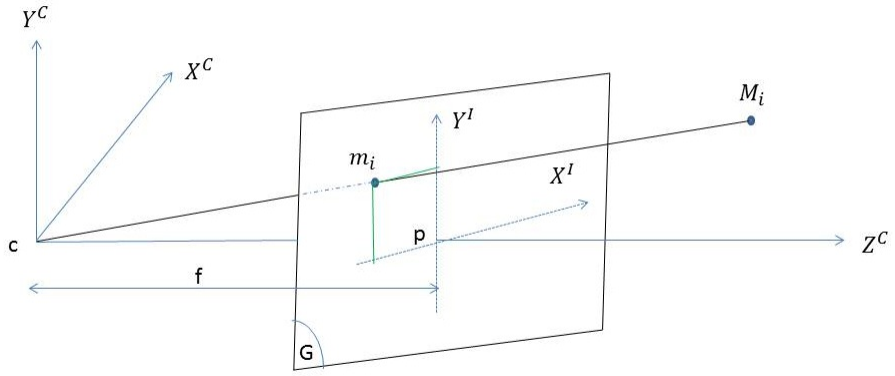


Figure 2.4: Camera and Image Plane Coordinate Frame (after: [29]): The point c is the origin of camera coordinate frame, or the camera center. The Z^C axis is called the principal axis or optical axis for the camera system.The principal point(p) is where the optical axis intersects the image plane. It establishes the origin of the image plane coordinate system.The image plane coordinate frame is defined by the origin p , the x-axis X^I, and the y-axis Y^I. The distance between the point c and p is called focal length(f).

from camera coordinate frame to image plane coordinate frame is:

$$\begin{bmatrix} x_i^I \\ y_i^I \end{bmatrix} = f/Z_i^C \times \begin{bmatrix} X_i^C \\ Y_i^C \end{bmatrix} \tag{2.7}$$

2.2.2 Camera Models

From a computer vision stand point and in order to work with a given camera, a camera model and several camera related parameters must be fixed and computed in advance.

Pinhole Camera Model

Also known as Camera Obscura model, the pinhole camera model represents the mathematical relationship between the coordinates of a $3D$ point and its projection onto the image plane. The basis of projection comes from a model where all light from a scene passes through a single point and projects an inverted image on the opposite side. Figure 2.4 illustrates a pinhole camera model where the image points are in front of the camera center. A $3D$ point is projected through origin C to the image plane (G) along Z^C the relationship between the $3D$ and $2D$ coordinates is given by Equation 2.7.

Camera Calibration

The purpose of camera calibration is to determine the intrinsic camera parameters that affect the imaging process. These parameters are:

- Position of image center on the image: commonly noted by (c_x,c_y) which are the coordinates of points p in Figure 2.4
- Different scaling factors for row pixels (s_x) and column pixels (s_y), i.e., pixels are not typically square
- Focal length: denoted by f having two scaled components f_x=f/s_x and f_y=f/s_y
- Lens distortion: represents the radial distortion coefficient

For a CCD camera the lens distortion is assumed to be zero. The corresponding camera calibration matrix is given by:

$$K = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.8}$$

In the current thesis the scaling factors are assumed to be equal to one and therefore $f = f_x = f_y$. There is a very large body of literature on the subject of camera calibration, mainly influenced by Roger Tsai's work [30]. These can be roughly classified into two main methods:

- conventional method: uses $2D$ and $3D$ data to compute camera parameters. may work with single view of image.
- camera self-calibration: only need image data for camera calibration but need image data from multiple views.

Several calibration toolboxes has been implemented in both Matlab and C as in [31]. This work has used a publicly-available calibration toolbox created by George Vogiatzis and Carlos Hernández and can be found on [32]. A detailed description of the implementation will further be discussed in Chapter 4.

2.2.3 Projection Models

Orthographic Projection

In an orthographic projection, a camera is at infinite distance from a $3D$ scene. All projection lines are parallel to optical axis $X_i^I = X_i^C$ and $Y_i^I = Y_i^C$ as seen in Figure 2.5 (because the focal length is neglected in front of the infinite distance). Orthographic projections are a small set of transformations often used to show profile, detail, or precise measurements of a $3D$ object.

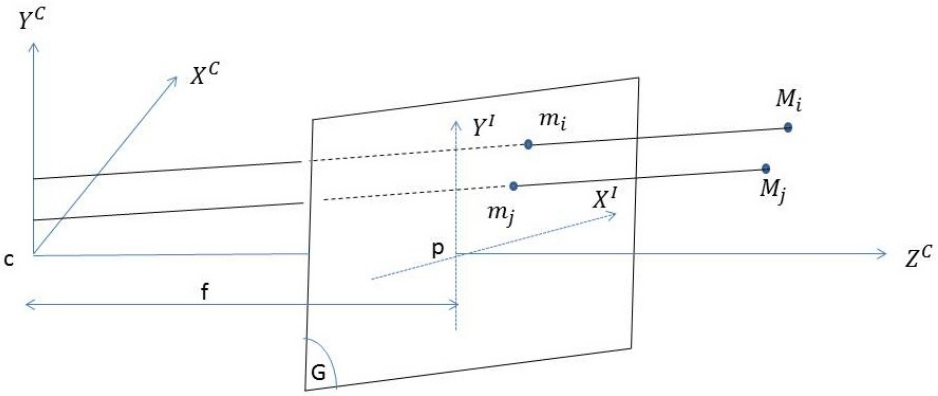


Figure 2.5: Orthographic Projection (after: [29]): In this projection camera is at infinite distance from the 3D world and all projections lines are parallel to the optical axis (Z^C)

Perspective Projection

A distant object in perspective projection appears smaller than objects nearby. In general, a camera frame is not aligned with the world frame. Image frame is not at image center, but it is still parallel to the camera frame. Perspective projection equations for point M_i (Figure 2.6) are

$$X_i^I = X_i^C f / Z_i^C \tag{2.9}$$

$$Y_i^I = Y_i^C f / Z_i^C \tag{2.10}$$

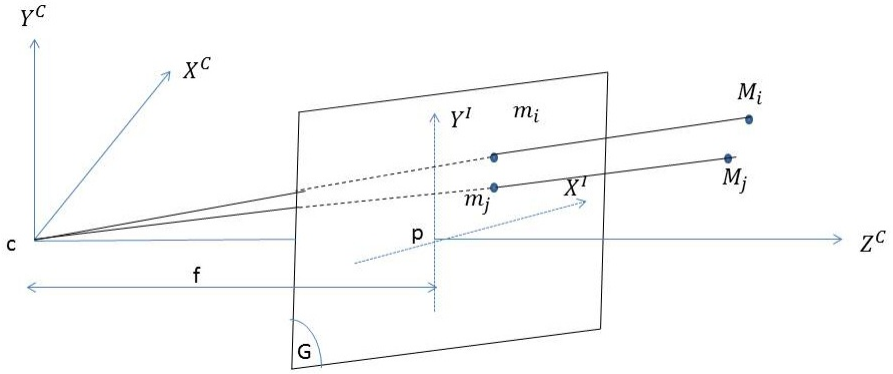


Figure 2.6: Perspective Projection (after: [29]): A 3D feature point M_i is projected onto an image plane (G) on point m_i with perspective rays originating at the center of projection (c), which would lie within the physical camera

.

Scaled Orthographic Projection

A Scaled Orthographic Projection (SOP) is a first order approximation of the perspective projection. It assumes that the variation in the direction of the optical axis is small compared to the viewing distance. In SOP, the image of a point M_i of an object is a point m_i of the image plane G which has coordinates:

$$X_i^I = X_i^C f / Z_0^C \tag{2.11}$$

$$y_i^I = Y_i^C f / Z_0^C \tag{2.12}$$

where Z_0^C is the z-coordinate of reference point M_0 in the camera coordinate frame, the ratio $s = f/Z_0$ is called the scaling factor of the SOP.

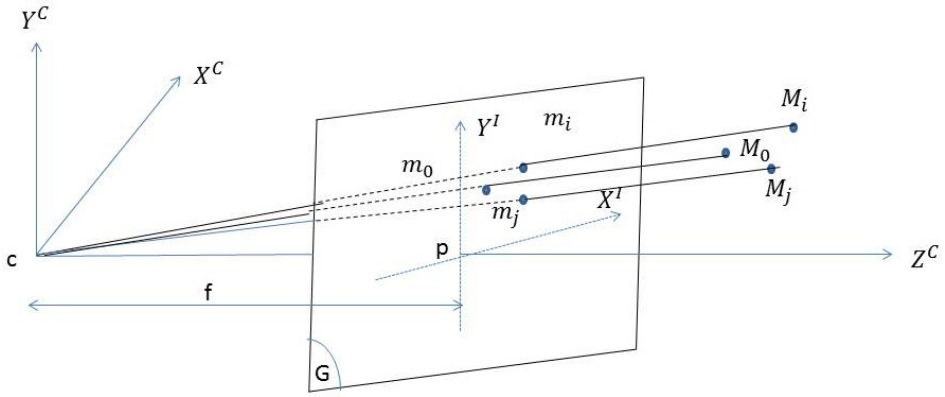


Figure 2.7: Scaled Orthographic Projection (after: [29]): An object is given by three points M_0 the reference point and M_i,M_j. The coordinates of all objects points are expressed in function of the Z coordinate of the reference point.

Camera Rotation and Translation

Rotations: Three basic rotations are performed separately [33]. These rotations are:

- Rotation along x-axis with angle ϕ also known as Roll

$$R_x(\phi) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi \\ 0 & \sin\phi & \cos\phi \end{pmatrix} \tag{2.13}$$

- Rotation along y-axis with angle θ also known as Pitch

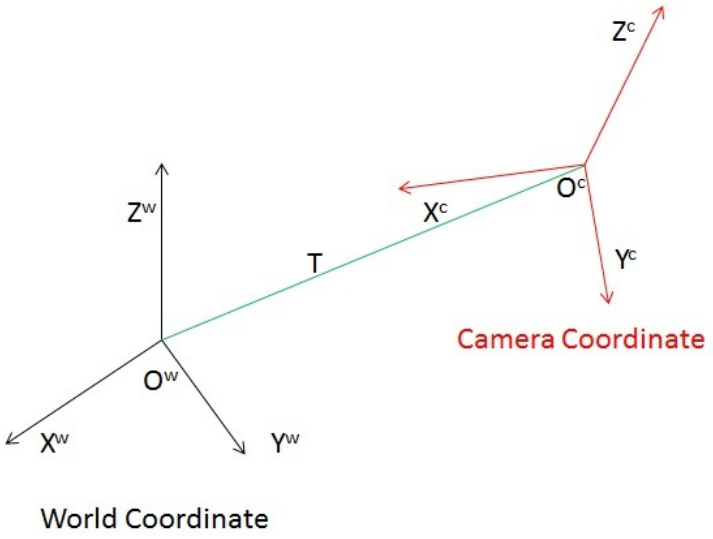


Figure 2.8: Camera Rotation and Translation

$$R_y(\theta) = \begin{pmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{pmatrix} \tag{2.14}$$

- Rotation along z-axis with angle ψ also known as Yaw

$$R_z(\psi) = \begin{pmatrix} \cos\psi & -\sin\psi & 0 \\ \sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.15}$$

The resulting overall rotation matrix is

$$R = R_z(\psi) \times R_y(\psi) \times R_x(\phi) = \begin{pmatrix} R_{11} & R_{12} & R_{13} \\ R_{21} & R_{22} & R_{23} \\ R_{31} & R_{32} & R_{33} \end{pmatrix} \tag{2.16}$$

where

$$
\begin{aligned}
R_{11} &= \cos\psi\cos\theta \\
R_{12} &= -\sin\psi\cos\phi + \cos\psi\sin\theta\sin\phi \\
R_{13} &= \sin\psi\sin\theta + \cos\psi\sin\theta\cos\phi \\
R_{21} &= \sin\psi\cos\theta \\
R_{22} &= \cos\psi\cos\phi \\
R_{23} &= -\cos\psi\sin\phi + \sin\psi\sin\theta\cos\phi \\
R_{31} &= -\sin\theta \\
R_{32} &= \cos\theta\sin\phi \\
R_{33} &= \cos\theta\cos\phi
\end{aligned}
$$

Translation Vector: The translation vector is a $3D$ vector composed of displacement along x-axis, y-axis, and z-axis

$$
T = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{2.17}
$$

CHAPTER 3:
Computer Vision Methods for UAV to UAV Detection

This chapter provides the theoretical foundation for the computer vision algorithms that will be used. The first section deals with the detection problem by presenting two approaches, where the former uses edge detection and image smoothing while the latter uses a morphological filtering and color-based detection approach. The second section of this chapter presents discussion on the implementation of these approaches and the experimental results.

3.1 UAV Detection Algorithms

3.1.1 UAV Detection using Edge Detection and Image Smoothing

Several steps are involved in this algorithm (Figure 3.1).

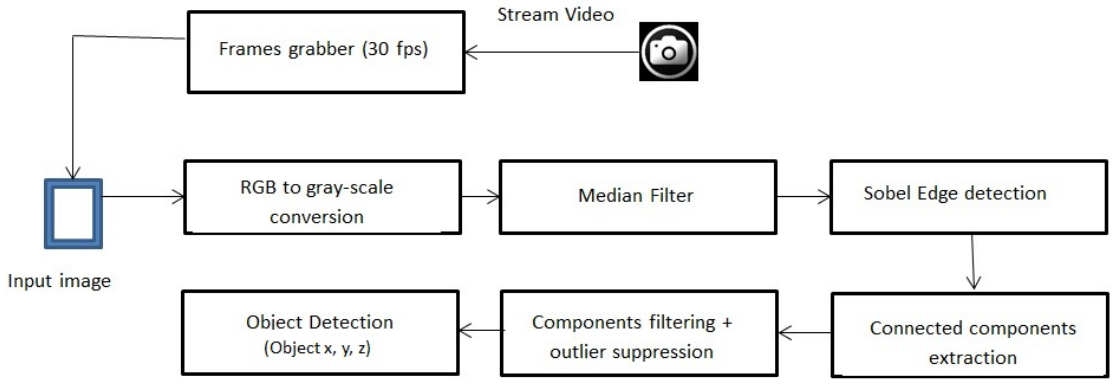


Figure 3.1: UAV Detection using Edge Detection and Image Smoothing

The first step in this approach is the "frame grabber," in which the input video is converted into frame sequences at a regular rate called Frames Per Second (FPS). The second step is "Red, Green, Blue (RGB) to Gray scale conversion." In an RGB image, a color pixel is described by a triple (red (R), green (G), and blue (B)) of color values. The purpose of this step is to map the color values to a single number, such as a gray scale value for intensity, to represent each pixel. For example, the intensity information is composed exclusively of shades of gray, varying from black at the weakest intensity to white at the strongest. Several methods have been used to perform this conversion to generate a single representative value for each pixel:

- The *lightness method* averages the most prominent and least prominent colors:

$$\frac{\max(\mathrm{R},\mathrm{G},\mathrm{B})+\min(\mathrm{R},\mathrm{G},\mathrm{B})}{2}$$

- The *average method* simply averages the three color values:

$$\frac{\mathrm{R}+\mathrm{G}+\mathrm{B}}{3}$$

- The *luminosity method* is a more sophisticated version of the average method [34]. It also averages the values, but it forms a weighted average to account for human perception. The formula for luminosity is:

$$0.21\mathrm{R}+0.71\mathrm{G}+0.07\mathrm{B}$$

The next step is "median filtering," which is a non-linear spatial filter widely used in image processing because it preserves useful detail in the image while reducing noise. This is a preliminary step to improve the later effort of edge detection. Roughly speaking, the median filter takes the median value of pixels in a predefined window [35].

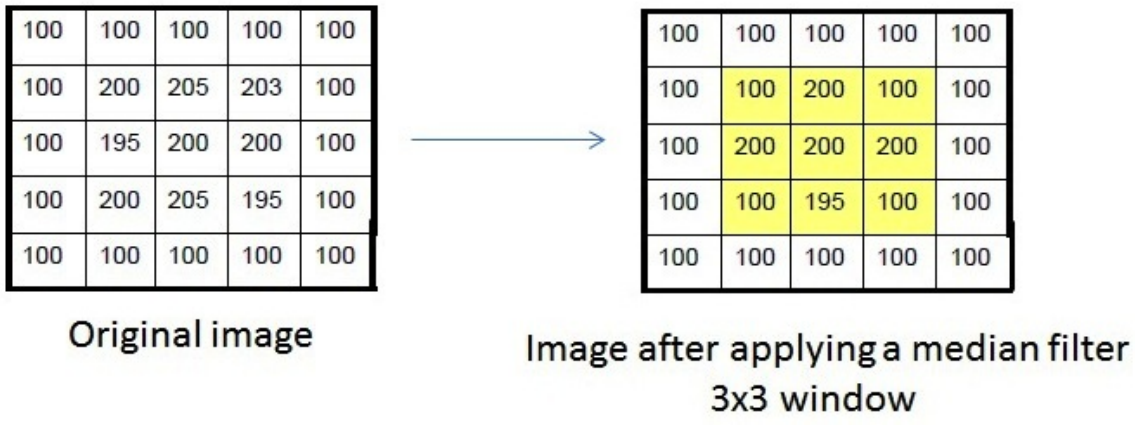


Figure 3.2: Median Filtering (from: [35]): The window size is 3×3 pixels. If we consider the pixel at coordinates $(2,2)$ in the image matrix which has a value equal to 200, its neighborhood values (inside the defined window) are: $100, 100, 100, 100, 205, 100, 195, 200$, for which the median value is 100 and replaces the first value (200).

After median filtering is performed, the frame is passed through Sobel edge detection. Edge detection is a fundamental operation in computer vision. Edges are significant local changes of intensity in an image and they typically occur on the boundary between two dif-

ferent regions in an image. Edge detection extracts salient features of the image and produces a line drawing of the image. The Sobel edge detector is a $2D$ spatial gradient measurement on an image. It uses a convolution kernel to create a series of gradient magnitudes so that it emphasizes regions of high spatial frequency that correspond to edges [36]. The Sobel edge detector uses a pair of 3×3 convolution kernels. An example of two associated kernels is:

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

The next step is to perform "connected components extraction and image labeling." In image processing, a "connected components" algorithm groups pixels into components based on pixel connectivity. That is, it finds regions of connected pixels that have the same value. Once all components have been determined, each pixel is labeled according to the unique identifier of its blob, identifying blob membership. Finally, a filtering operation is performed to eliminate outliers and non-persistent features and to isolate and detect the object. The resulting output is the center location of the detected object, usually represented in the image plane.

3.1.2 Morphological Filtering and Color-Based Detection

This algorithm is based on the use of a finite number of color markers attached to the UAV, as in Figure 2.1 with six markers. They are spatially placed in an asymmetric manner that helps facilitate robust detection and, as will be seen in later in this thesis, assist in accurate pose estimation for planar points. The goal here is to detect these six colored patches and determine their positions in the image plane.

Color Recognition

In order for a computer to process images acquired by external electro-optical sensors, the image is encoded using different colors spaces. A color space describes the range of colors that the camera can see. The color space is used to represent each color pixel in an image. For this thesis, we introduce the two main color spaces used in computer vision, which are the RGB-space and the HSV-space.

- **RGB Color Space**: The RGB color space [37] uses additive color mixing; it is "all possible colors" that can be made from three colorants for red, green and blue. It describes what kind of light needs to be emitted to produce a given color. In the RGB color space,

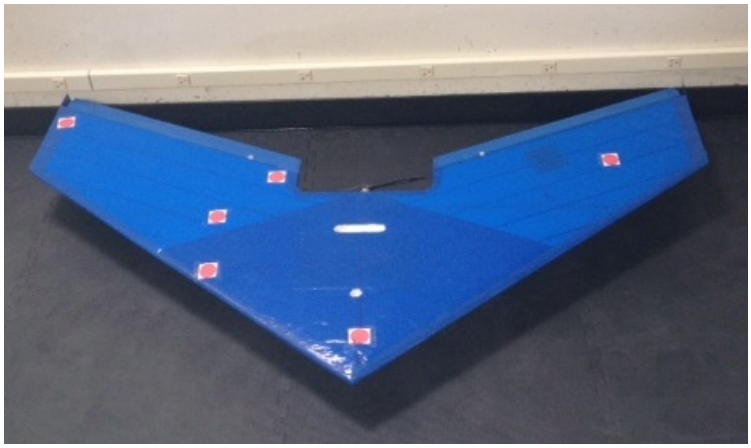


Figure 3.3: Light Beacon Design: This figure depicts six red markers placed in an asymmetric manner on the UAV's surface. These markers are coplanar as we are dealing with coplanar case

individual values for red, green and blue are stored as described in thevus section. A common notation for RGB pixels is $I(x, y, band)$ where I is the image; x, y are the x and y pixel coordinates, and *band* is the color channel, either one for red, two for green or three for blue.

- **HSV Color Space**: The HSV (hue, saturation, value) color space [37], also known as HSB (hue, saturation, brightness), is often used by artists because it is generally more natural to think about a color in terms of hue and saturation rather than in terms of additive or subtractive color components. The HSV-space is a transformation of the RGB colorspace, and its components and colorimetry are relative to the RGB colorspace from which it was derived.

Detection Process

Here, a colored image of three channels, that is, R, G, and B, is taken. Thus, each pixel has a red, green and blue value. The aim is to retrieve these values from an image, then extract and isolate specifically the red patches. (Red is selected arbitrarily and the described approach extends naturally to other colors.) In an ideal case, a red pixel means R=255, G=0, and B=0. However, in reality, there will also be green and blue components, too. In order to do segmentation according to color, the following three-step algorithm are applied, as developed in open source [38]:

Thresholding: Image thresholding is a basic technique used for image segmentation. The main purpose is to extract the object of interest from the background. That is, it is necessary to select a threshold δ that separates these two image components. If we consider that the binary image

is represented by $I(x,y)$ (neglecting *band*), then those pixels (x,y) for which $I(x,y) \geq \delta$ are considered points representing the object.

Morphological Filtering: Morphological filtering [39] is based on two primary morphological operations known as *dilation* and *erosion*. The dilation of a gray scale image, $I(x,y)$, by a morphological structuring elements, $SE(x,y)$, is defined by Equation 3.1(from: [40]), where $\oplus$ denotes the dilation operator and p is a pixel in the image I:

$$I \oplus SE = \bigcup_{p \in I} SE_p \tag{3.1}$$

Dilation is used for filling gaps in images; it gradually adds more individual pixels to the output image when sweeping I with SE (Figure 3.4). That is, it grows image regions. For each pixel in the image I we superimpose the structuring element SE on top of the image matrix so that the origin of the structuring element coincides with the current pixel position. If the origin of SE matches the underneath pixel in I, the contents of SE are copied to the resultant image.

In contrast, the erosion of a gray scale image, $I(x,y)$, by morphological structuring elements, $SE(x,y)$, is denoted by the $\ominus$ operation and its definition is given by Equation 3.2 (from: [40]), where p is a pixel in the image I and s is a pixel in the structuring element SE:

$$I \ominus SE = \{p | p + s \in I \forall s \in SE\} \tag{3.2}$$

Erosion is used to eliminate unwanted noise in a processed image by shrinking image regions (Figure 3.4). For each pixel in the image I, the structuring element SE is superimposed on the image matrix, so that the origin of the structuring element coincides with the current pixel position. If every pixel underneath the structuring element coincides with the pixel of SE, the current pixel is considered a foreground pixel and set to one; otherwise, the pixel is considered a background pixel and set to zero.

The combination of these two basic operations form *morphological opening* and *morphological closing*. The morphological opening is an erosion followed by a dilation (Figure 3.5), using the same structuring element SE for both operations. The opening operation has basically the same effect as erosion, but it is less destructive in the sense that it tends to preserve foreground pixels that have a similar shape as SE.

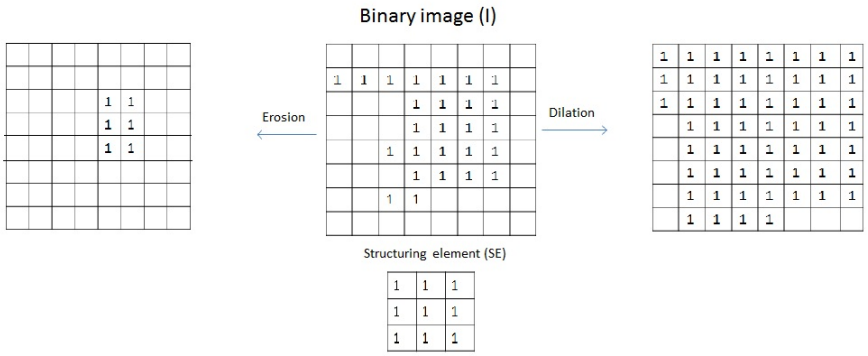


Figure 3.4: Basic morphological operations (after: [39]): The figure illustrate a binary image I to which we apply a morphological erosion and dilation using the structuring element SE.

On the other hand, the morphological closing is a dilation followed by an erosion (Figure 3.5), using the same structuring element SE for both operations. It is similar to dilation, but it is less destructive in a sense that fills in holes and narrows valleys along edges in the image, I.

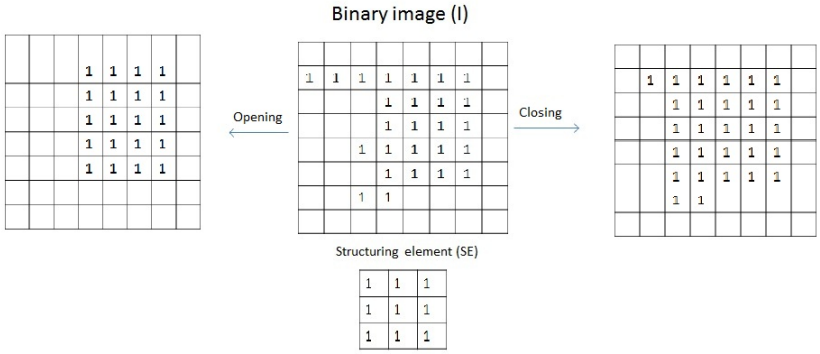


Figure 3.5: Advanced morphological operations (after: [39]): A dilation followed by an erosion gives a morphological closing, and an erosion followed by dilation gives morphological opening using the same structuring element SE.

Blob Masking and Filtering Process: After morphological filtering, the next step is "blob masking and filtering process," where the resulting image form the previous steps is used to fill background holes, for example, by using the Matlab function `imfill`. A mask for red color is then created, so that when applied, the masked image extracts the red blobs with a black background (that is, non-masked pixels are set to zero). A filtering step is needed to suppress residual noise from the previous operations. The last step is to compute the centroid of the individual red blobs in the image coordinate frame (Figure 3.12).

3.2 Implementation and Experimental Results

3.2.1 Detection Metrics

Many measures of detection performance that are commonly used can readily be found in the literature, including:

- High correct (positive) detection rate
- Low false alarm rate
- Real-time operation measured by computation time (of algorithms)

3.2.2 UAV Detection using Edge Detection and Image Smoothing

To evaluate and test this approach video of one approaching plane on *USS JOHN C. STENNIS* (CVN-74) was used, courtesy of previous work done by Ryan S. Yusko [39].

As mentioned in Section 3.1.1, the first step is to perform the frame grabbing. The approach video was stored in MPEG-2 format with a resolution of 720 pixels wide by 480 pixels high, interlaced at 29.97 FPS [39]. We developed a Matlab function to read the video frame by frame and save each frame as a JPEG image. Figure 3.6 illustrates the collection of grabbed frames.

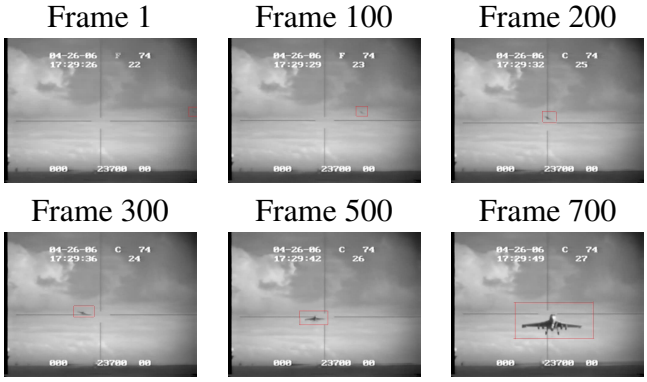


Figure 3.6: Frame Grabbing: Illustration of frame grabbing that takes as input the test video and converts it to individual frames at near 30 frames per seconds.

The second step is "frame cropping." After some experiences with these frames, we noted that the annotations that show on the video make the detection difficult, so it was decided to define a Region of Interest (ROI) as being the upper right quadrant of the image frame to keep only the ROI (Figure 3.7). The "cropping function" developed in Matlab can be found in the Appendix.

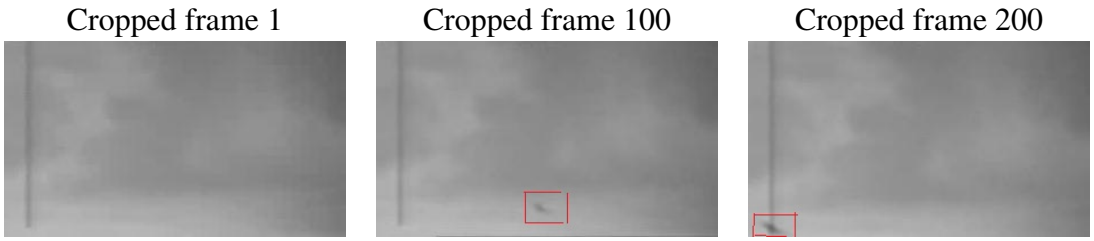
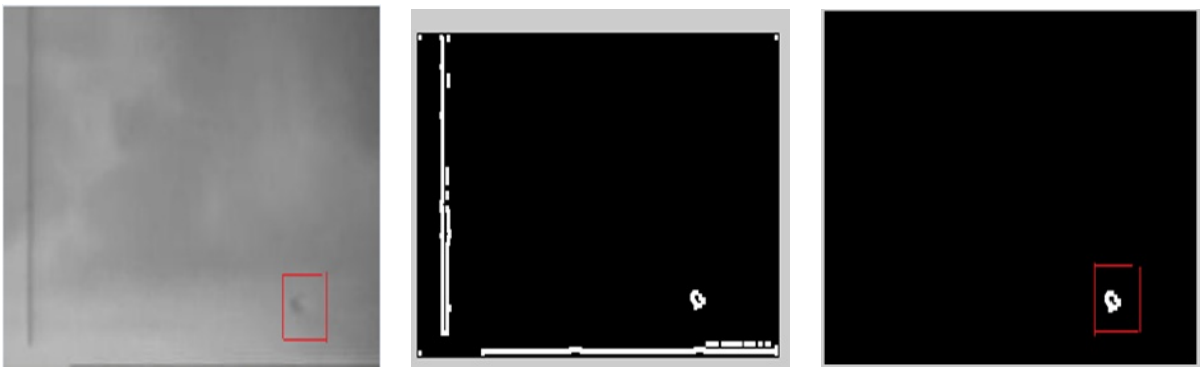


Figure 3.7: Frame cropping: It has been noted that most of time the approaching plane is on the upper right corner where both x and y coordinate are positive, so a cropping operation was needed eliminate the digital numbers that comes with the recorded video, which makes the detection harder.

For illustration purposes, we chose frame number 20 for application of, respectively, RGB to gray scale conversion, Sobel detection to detect connected components, and filtering and outlier suppression as outlined previously. The sequence of the process is illustrated in Figure 3.8.



(a) (b) (c)

Figure 3.8: Detection using Edge Detection and Image Smoothing, showing application of (a) RGB to gray scale conversion, (b) connected components detection, and (c) filtering and outliers suppression

Detection Rate Results

We performed object detection experiments using the edge detection and image smoothing approach over 191 frames, with the following results:

Number of tested frame	Detection	Missed detection	Mean execution time [sec]
191	169	22	1.4

Table 3.1: Detection rates: The probability of detection is 0.884 and the probability of missed detection is 0.115

Indeed, this approach shows high detection rate with a detection probability of 0.884 and a mean execution time of 1.4 seconds. The mean execution time is the average computation time

measured after running the algorithm on 191 frames with a HP laptop Intel Core i5 and Matlab as programming environment. The assumption under which this approach was implemented is that there are no other objects on the scene having the same size as the target UAV. That is, this approach exhibits high false alarm rates in highly cluttered image environments where many objects appear to have in the same shape as the target UAV. Investigation of methods to address this issue are reserved for future work.

3.2.3 Morphological Filtering and Color-Based Detection

This approach is mainly inspired by the work in [38]; however, that study used HSV as color space, while the present study use RGB. Though the emphasis in this section is to describe the approach as applied to UAV detection, its use is further motivated by our intent to apply it to the pose estimation problem in Chapter 4. Six red patches are used in an asymmetric configuration on the UAV plane as previously shown in Figure 3.3. This method uses a morphological structuring element having a disk shape with a radius = 2 pixels using the Matlab function call (see also Appendix):

```
structuringElement = strel('disk', 2)
```

Then the morphological closing operation is applied to the image to enlarge the boundaries of foreground (bright) regions and shrink background color holes in such regions using the following Matlab function:

```
redObjectsMask = imclose(redObjectsMask, structuringElement)
```

The next step in this approach is to determine the color threshold (Figure 3.10). using the individual color bands, as illustrated in Figure 3.9. After determining the upper and lower bound of each color band's threshold, these thresholds are applied to mask green and blue colors, while keeping the red. These thresholding steps provide the input to the morphological filter, namely the image showing only red objects, as illustrated in Figure 3.11.

The next step in the detection process extracts connected components (red markers) and determines their center of mass locations in the image plane coordinate frame, as illustrated in Figure 3.12. These locations are then recorded and used to register detection of the UAV.

Detection Rate

The metric for the presented detection algorithm is the number of red spots detected. That is, if more than four red patches are successfully detected, the detection process for localizing the UAV object within an image is considered successful. To test this algorithm, we generated a

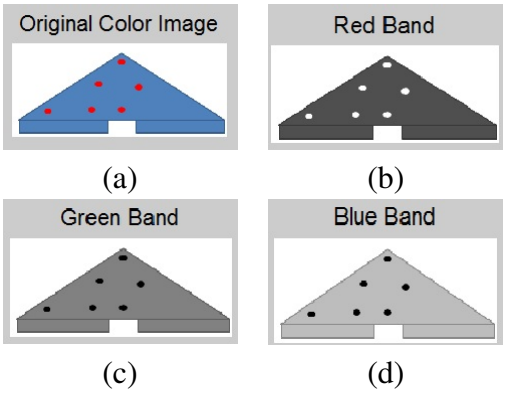


Figure 3.9: Color Band Extraction: Individual color bands of red, green, and blue (panels (b), (c), (d), respectively) were extracted and saved, from the original image shown in (a).

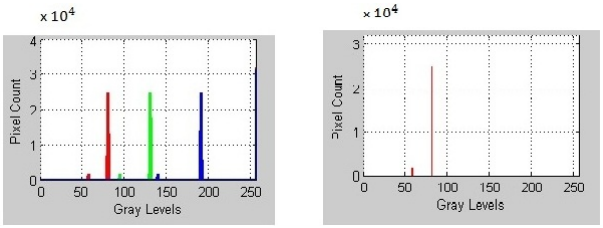


Figure 3.10: Threshold Extraction: Determine a threshold for each color, e.g., by use of color histograms: (a) histogram for all three color bands, (b) histogram associated only with the red band.

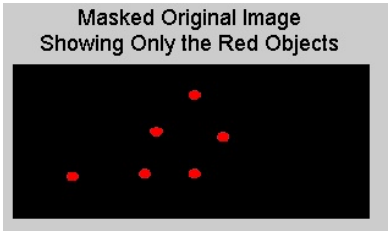


Figure 3.11: Red Color Detection: The individual color thresholds are applied to mask green and blue colors, while keeping the red. The resulting image is shown, highlighting the locations of the six red patches placed on the UAV body.

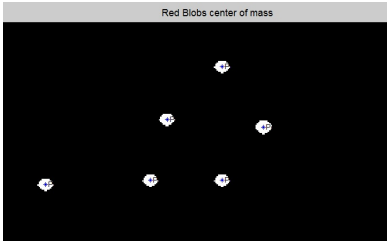


Figure 3.12: Color blobs' Centers of Mass: The center of gravity for each individual red blob is determined and saved. Their coordinates are computed on the image plane coordinate frame.

large number of images in laboratory settings using different configurations of red patches on the UAV, and obtained the following results.

Number of runs	Detection	Missed detection	Mean execution time (seconds)
40	38	2	2

Table 3.2: Detection rates: Where "number of runs" are the number of randomly generated red color markers in a planar distribution. In this approach the probability of detection was 0.95 and the probability of missed detection was 0.05

Indeed, this approach shows high detection rate with a detection probability of 0.95 and a mean execution time of 2 seconds measured after running the algorithm on 40 runs with a HP laptop Intel Core i5 and Matlab as the computation environment. The assumption under which this approach was implemented is that there are no other red objects on the scene and the lighting conditions are suitable for detection red color. That is, this approach exhibits high false alarm in the presence other sources of red color and in a poor lighting conditions. Robustness studies are left for future study.

3.3 Summary of Results

The first detection method "UAV detection using edge detection and image smoothing" shows high detection rate with a detection probability 0.884 and a mean execution time of 1.4 seconds, which means the average computation time measured after running it on 191 frames with a HP laptop Intel Core i5 and Matlab as the computation environment. This approach was implemented under the assumption that there are no other objects in the scene having the same size as the target UAV. That is, it exhibits high false alarm in highly cluttered sky environments where many objects appear to be the same shape as the target UAV. A tracking function can be used to help eliminate these false alarms.

The second detection method "morphological filtering and color-based detection" where red color markers are placed on the surface of the target UAV in an asymmetric configuration. This method shows high detection rate with a detection probability 0.95 and a mean execution time of 2 seconds, which means the average computation time measured after running it on 40 random red patch distribution. This approach was implemented under the assumption that the target UAV is detected if it is possible to detect four red patches. That is, it exhibit low detection rate in low lighting condition and the target UAV orientation with respect to the camera. It also faces false alarm problem when there are other sources of red color.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 4:
Computer Vision Methods for UAV Pose Estimation

This chapter addresses the UAV pose estimation problem. The first section presents a theoretical foundation for three pose estimation methods including one that provides pose estimation using planar homography, a non-iterative pose estimation approach developed by researchers at EPFL, and application of the Posit algorithm for coplanar points to estimate pose. The second section presents their implementation and experimental results, followed by a summary of results.

4.1 Pose Estimation Methods

The goal of pose estimation is to get the position and orientation of an object (i.e., six degrees of freedom) from a $2D$ image (e.g., from a CCD camera). That is, given a calibrated camera, an object with identifiable feature points, and their corresponding points on the image plane, the goal is to obtain the rotation (*roll*, pitch, *yaw*) and the translation (x, y, z) of the object relative to a known reference frame, as illustrated in Figure 4.1. As we mentioned in Chapter 1, this study focuses on *planar* feature points, as the UAV model used has a nearly planar shape.

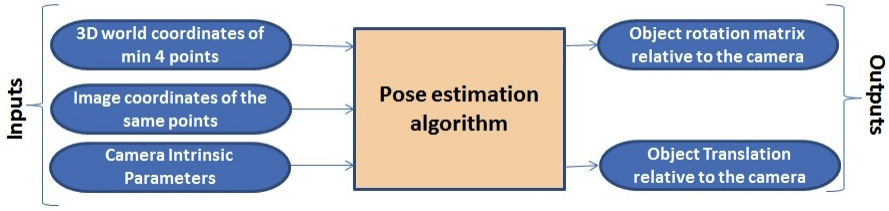


Figure 4.1: Pose Estimation Process: A generic pose estimation algorithm takes as input the $3D$ features points, the image coordinates of these features points on the image plane, and the camera intrinsic parameters and provides as output the rotations angles and the translation vector.

4.1.1 Pose Estimation using Planar Homography

The technique used herein follows from the method proposed by Yang and his colleagues [27], although pose estimation using homographies has extensive literature with applications including finding ground planes, walls, and roads.

Problem Formulation

Suppose there are N object points represented by its homogeneous coordinates in the UAV reference frame, where the i^{th} point is represented by coordinates, $M_i = (X_i^B, Y_i^B, Z_i^B, w_i)^T$. Recall that the superscript B denotes the (UAV) body-fixed frame. The corresponding homogeneous

coordinates of their images with respect to the image plane reference frame are denoted by superscript I, such that each i^{th} point is represented by $m_i = (x_i^I, x_i^I, \lambda_i)^T$. The terms w_i and λ_i are the homogeneous parameters and for simplicity, are usually set to one. The geometry and associated reference frames are illustrated in Figure 4.2. Taking into account that all points are coplanar points, the UAV body frame is adjusted in a way that the plane Π, the plane that contains all the feature points, coincides with the plane $OX^B\ Y^B$ which will let $Z_i^B = 0$ for all object points. Given these definitions, the objective is to compute the homography, H, from which it is possible to extract the rotation and translation matrices, R and T, respectively.

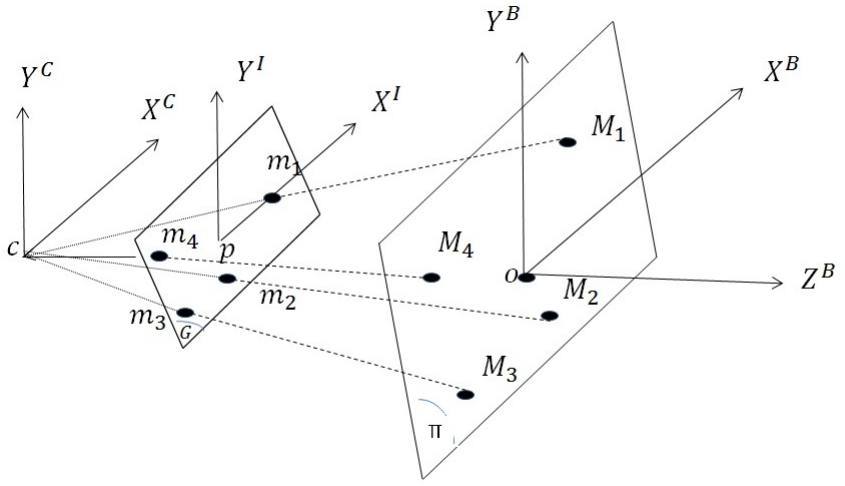


Figure 4.2: Perspective Projection (after: [27]): A set of features points $\{M_1, M_2, M_3, M_4\}$ are projected into the image plane using perspective projection resulting in four image points $\{m_1, m_2, m_3, m_4\}$ having their coordinates represented on the image plane reference frame.

The points M_i and their perspective projections m_i are related by Equation 4.1:

$$m_i = KR \begin{bmatrix} I_{3\times 3} & T \end{bmatrix} M_i \tag{4.1}$$

where K is the 3×3 camera calibration matrix (c.f. Chapter 2, Equation 2.8) and R is 3×3 rotation matrix that represents the rotation of the object relative to camera coordinate frame and T is 3×1 translation vector. M_i is 4×1 homogeneous coordinate of object points i, while m_i is 3×1 homogeneous coordinate of the corresponding image points. The augmented matrix $\begin{bmatrix} I_{3\times 3} & T \end{bmatrix}$ is a 3×4 matrix formed by concatenating a 3×3 identity matrix and the 3×1 vector, M_i.

Equation 4.1 can be rewritten as

$$m_i = K \begin{bmatrix} R_1 & R_2 & RT \end{bmatrix} \hat{M}_i \tag{4.2}$$

$$m_i = H\hat{M}_i \tag{4.3}$$

where $\hat{M}_i = (X_i^B, Y_i^B, w_i)^T$, which is equivalent to point M_i without the z-component. R_1 and R_2 are the first two columns of the rotation matrix R. As shown in [27], the matrix $\begin{bmatrix} R_1 & R_2 & RT \end{bmatrix}$ is invertible and as a consequence, defines a planar homography, H, which relates m_i and $\hat{M}_i$, defined as

$$H = K \begin{bmatrix} R_1 & R_2 & RT \end{bmatrix} \tag{4.4}$$

Using the fact that the calibration matrix K is known (from prior calibration or from provided specifications), we can multiply both sides of Equation 4.2 by the inverse, K^{-1}. The product $K^{-1}m_i$ is denoted by m_i' resulting in the following equation:

$$m_i' = K^{-1}m_i = \begin{bmatrix} R_1 & R_2 & RT \end{bmatrix} \hat{M}_i = H'\hat{M}_i \tag{4.5}$$

where K^{-1} is the inverse of matrix K and $m_i' = (u_i, v_i, \lambda_i)^T$. The transformed homography matrix, H', (with elements h_{ij}) is given by

$$H' = K^{-1}\,H \quad = \quad \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \tag{4.6}$$

Note that m_i' is equal to $H\hat{M}_i$ up to a scaling factor; therefore their cross product is zero, i.e., $m_i' \times H\hat{M}_i = 0$ where $\times$ represents the cross product of two vectors. This planar constraint can be rewritten as a homogeneous linear system of equations for each point m_i' and

M_i according to

$$A_i h = \begin{pmatrix} 0_{1\times 3} & -\lambda_i \hat{M}_i^T & v_i \hat{M}_i^T \\ \lambda_i \hat{M}_i^T & 0_{1\times 3} & -u_i \hat{M}_i^T \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ h_{13} \\ h_{21} \\ h_{22} \\ h_{23} \\ h_{31} \\ h_{32} \\ h_{33} \end{pmatrix} = 0 \tag{4.7}$$

where $0_{1\times 3}$ denotes a zero row-vector, $\hat{M}_i^T$ is the transpose of $\hat{M}_i$, $A_i \in R^{2\times 9}$ is defined for each point i, and $h \in R^{9\times 1}$ is the column vector comprising elements of the scaled homography matrix, H'. For n object points, the augmented matrix $A \in R^{2n\times 9}$ for all points represents a system of linear equations

$$A\ h = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_N \end{pmatrix} h = 0,$$

for which singular value decomposition can be used to solve for the solution, h. The scaled homography matrix, H', can be reconstructed from reshaping the column vector h.

The desired homography matrix, H, can now be computed by normalizing H'. The Euclidean distance was used for normalization purpose.

$$H = H' / \mathtt{norm}\left(H'_1\right) \tag{4.8}$$

The final steps include using the homography matrix, H, to determine the object's relative rotation and translation matrices. The first and second columns of H will be (after being converted to unit vectors) the first and second columns of the rotation matrix R as defined in Equation 4.4. The third column in the rotation matrix can be computed as the cross product of the first two columns, such that

$$R = \begin{pmatrix} H_1 & H_2 & H_1 \times H_2 \end{pmatrix} \tag{4.9}$$

The translation vector, T, can be computed using the following expression, where $H_3 = H_1 \times H_2$:

$$T = -R^T H_3 \tag{4.10}$$

4.1.2 Non-iterative Pose Estimation

A group of researchers from the École Polytechnique Fédérale de Lausanne (EPFL) proposed a non-iterative solution approach (referred to as the "EPFL algorithm" in this thesis) to the "Pose from n Points" (PnP) problem. They showed that their solution's computational complexity grows only linearly with n, the number of features points, and possesses the advantage of applying to both planar and non-planar feature point configurations.

Formulation

Recall that in the previously presented approach, it is assumed that the camera intrinsic parameters given by its calibration matrix K are known, and a set of n correspondences between $3D$ reference points (a.k.a, object points) and their $2D$ projections is available. The main idea behind the EPFL approach presented in this section is to define the coordinates of the n reference points as a linear combination of four known locations called *control points*. Denote by M_i, for $i = 1, 2, ..., n$ the i^{th} reference point in a general reference frame, and the j^{th} control point by C_j, for $j = 1, 2, 3, 4$.

Using the idea of the four control points as the bases for each reference point, the point coordinates of each M_i in the UAV body coordinate frame (denoted by superscript B) can be expressed as follows:

$$M_i^B = \sum_{j=1}^{4} \alpha_{ij} C_j^B, \qquad \text{with} \quad \sum_{j=1}^{4} \alpha_{ij} = 1 \tag{4.11}$$

where the α_{ij} are the normalized barycentric coordinates, which are uniquely defined for a given point M_i [28]. In the same way, the reference points in the camera coordinate frame (superscript c) are expressed as

$$M_i^c = \sum_{j=1}^{4} \alpha_{ij} C_j^c \tag{4.12}$$

It remains to define the coordinates of the control points such that the above definitions in Equations 4.11 and 4.12 hold. The stability of the proposed approach has been shown in the

literature to be improved by taking the centroid of the n reference points as one of the control points, and then selecting the remaining three control points such that they form a basis aligned with the principal directions of the reference points.

Recalling the camera calibration matrix, K, the $2D$ projections of the n reference points $M_i,\ i = 1,2,...,n$ are represented by

$$w_i \begin{pmatrix} u_i \\ v_i \\ 1 \end{pmatrix} = KM_i^c = K \sum_{j=1}^{4} \alpha_{ij} C_j^c \tag{4.13}$$

where w_i are scalar projective parameters, the projections in $2D$ are denoted u_i and v_i, and the elements of the control point coordinate vector are:

$$C_j^c = \begin{pmatrix} x_j^c \\ y_j^c \\ z_j^c \end{pmatrix}.$$

Notice from Equation 4.13, combined with the definition of K in Equation 2.8, that $w_i = \sum_{j=1}^{4} \alpha_{ij} z_j^c$. Substitution of this expression in the first two rows of Equation 4.13 results in the following two linear equations:

$$\sum_{i=1}^{4} \left(\alpha_{ij} f_u x_j^c + \alpha_{ij} (c_x - u_i) z_j^c \right) = 0 \tag{4.14}$$

$$\sum_{i=1}^{4} \left(\alpha_{ij} f_v y_j^c + \alpha_{ij} (c_y - v_i) z_j^c \right) = 0 \tag{4.15}$$

If these two linear equations are concatenated for all n reference points, the resulting homogeneous linear system can be represented as $Ax = 0$, where the solution is given by $x \in R^{12\times 1}$:

$$x = \begin{pmatrix} C_1^c \\ C_2^c \\ C_3^c \\ C_4^c \end{pmatrix}$$

which comprises the camera coordinates of each control point, and A is a $2n \times 12$ matrix. That

is, the solution for the homogeneous linear system belongs to the kernel of matrix A; therefore it is expressed as

$$x = \sum_{l=1}^{\dim \mathcal{N}\left(A^T A\right)} \beta_l v_l \tag{4.16}$$

where $\dim \mathcal{N}\left(A^T A\right)$ is the dimension of the null-space of $A^T A$, and v_l are the right-singular vectors of A corresponding to the singular values of A, from which the coefficients, β_l, can be determined. A more detailed description of this method and its derivation can be found in [28].

4.1.3 Pose Estimation using Posit Algorithm for Coplanar Points

DeMenthon and Davis [25] developed a method called "Pose from Orthographic Projection and Scaling with Iterations," also known as the PosIt algorithm. The method can estimate a pose of an object from a single image provided that four or more point correspondences between $3D$ and $2D$ are given. Two main parts involved in this method are:

- POS (i.e., Pose from Orthography and Scaling), which uses scaled orthographic projection to approximate the perspective projection and finds the rotation matrix and translation vector from a linear system of equations, and
- PosIt (POS with Iteration), which iteratively uses a scale factor for each point to enhance the found orthographic projection and then uses POS on the new points instead of the original ones until a threshold is met.

Notation

Figure 4.3 depicts the classic pinhole camera [25] with the following parameters:

- The camera center of projection is the point C
- The image plane G at a distance f (focal length) from C
- CX^C and CY^C axis points along the rows and columns of camera sensor, the CZ^C points along the optical axis. Their unit vectors noted $\hat{i}$, $\hat{j}$, $\hat{k}$, respectively.

It also shows the UAV body coordinate frame having its center at point M_0, called the reference point, and its three axes, M_0X^B, M_0Y^B , and M_0Z^B. The plane K is the plane parallel to image plane G passing through M_0. The images of the feature points M_i are called m_i with (x_i, y_i) as image coordinates.

Problem Definition

The purpose is to compute the rotation matrix R and the translation vector of the UAV coordinate frame. The rotation matrix is given by:

$$R = \begin{bmatrix} i_x^B & i_y^B & i_z^B \\ j_x^B & j_y^B & j_z^B \\ k_x^B & k_y^B & k_z^B \end{bmatrix} = \begin{bmatrix} i^{BT} \\ j^{BT} \\ k^{BT} \end{bmatrix} \tag{4.17}$$

where the matrix rows are the coordinates of the unit vectors $\hat{i}$, $\hat{j}$, and $\hat{k}$ of the camera coordinate frame as expressed in the UAV coordinate frame. In order to compute R, it is only necessary to compute vectors $\hat{i}$ and $\hat{j}$ in the UAV coordinate frame. The vector $\hat{k}$ can be computed as the normal vector found by the cross-product $\hat{i} \times \hat{j}$, referring to Figure 4.3.

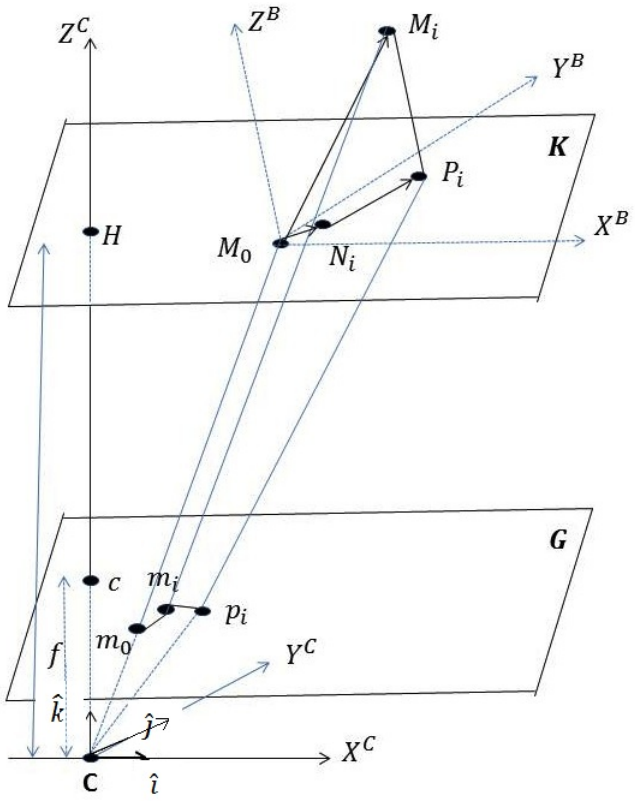


Figure 4.3: PosIt Projections (after: [25]): This figure illustrates a feature point M_i as well as the reference point M_0. The perspective projection of M_i is the point m_i, and its SOP is the point p_i. The point M_0 has the same image m_0 in SOP and perspective projections.

The translation vector is the vector $C\,M_0$ between the center of projection, C, and the

reference point M_0. This vector is aligned with $C\ m_0$ and equal to

$$Z_0/fCm_0 \tag{4.18}$$

so the translation is fully determined if Z_0 is computed. Each object point M_i is projected onto point m_i on the image plane, which leads to the following equalities:

$$x_0 = fX_0/Z_0 \tag{4.19}$$
$$x_i = fX_i/Z_i \tag{4.20}$$

for x coordinates and similarly for y coordinates:

$$y_0 = fY_0/Z_0, \tag{4.21}$$
$$y_i = fY_i/Z_i. \tag{4.22}$$

Equation 4.20 can be expanded into

$$x_i = f\frac{M_0Mi\cdot\hat{i}+X_0}{M_0Mi\cdot\hat{k}+Z_0} \tag{4.23}$$
$$= \frac{(f/Z_0\times M_0Mi)\cdot\hat{i}+x_0}{(1/Z_0\times M_0M_i)\cdot\hat{k}+1} \tag{4.24}$$

where the second equation is found by dividing numerator and denominator by Z_0.

It has been shown [25] that a necessary and sufficient condition for a pose defined by i, j, x_0, y_0, and Z_0 to be exact pose is that these parameters satisfy, for all points M_i, the equations:

$$M_0Mi\cdot I = x_i(1+\varepsilon_i) - x_0 \tag{4.25}$$
$$M_0Mi\cdot J = y_i(1+\varepsilon_i) - y_0 \tag{4.26}$$

where

$$J = f\times\hat{j}/Z_0 \tag{4.27}$$
$$I = f\times\hat{i}/Z_0, \text{and} \tag{4.28}$$
$$\varepsilon_i = (1/Z_0)M_0M_i\cdot\hat{k}. \tag{4.29}$$

The terms $x_i(1+\varepsilon_i)$ and $y_i(1+\varepsilon_i)$ are the coordinates $x_i^{'}$ and $y_i^{'}$ of the point p_i, which are the scaled orthographic projections of features points M_i (refer to Figure 4.3). The z coordinate of the vector M_0Mi is the dot product $M_0Mi.kZ_i - Z_0$, so

$$(1+\varepsilon_i) = 1 + (Z_i - Z_0)/Z_0 \tag{4.30}$$

$$= Z_i/Z_0 \tag{4.31}$$

where the second expression is found using Equation 4.20. Since p_i is the perspective projection of the point P_i that has the same x-coordinate as M_i and a z-coordinate equal to Z_0. If we substitute the value of x_i found on Equation 4.20, the x-coordinate x_i' of p_i is precisely determined.

$$x_i' = x_i((1+\varepsilon_i) = x_i Z_i/Z_0 \tag{4.32}$$

$$x_i' = fX_i/Z_0 \tag{4.33}$$

Finding Poses

Equation 4.25 and Equation 4.26 provide a linear systems of equations in which the only unknown are I and J for a given values of ε_i . Solving for I and J leads to i and j by normalization and Z_0 will be the norm of I. This algorithm is called POS (Pose from Orthography and Scaling) which provides an approximation of the pose. But once i and j have been computed, more exact values can be recomputed using Equation 4.26 solving the linear system again. This iterative application of POS is called PosIt (POS with Iterations).

Solving the System of Equations

The dot product in Equations 4.25 and 4.26 can be expressed in terms of vector coordinates in the UAV coordinate frame.

$$\begin{bmatrix} X_i^B & Y_i^B & X_i^B \end{bmatrix} \begin{bmatrix} I_x^B & I_y^B & I_z^B \end{bmatrix}^T = x_i(1+\varepsilon_i) - x_0 \tag{4.34}$$

$$\begin{bmatrix} X_i^B & Y_i^B & X_i^B \end{bmatrix} \begin{bmatrix} J_x^B & J_y^B & J_z^B \end{bmatrix}^T = y_i(1+\varepsilon_i) - y_0 \tag{4.35}$$

Writing the last two equations for the n objects points results in the following two linear systems:

$$A\ I = x'$$
$$A\ J = y'$$

The matrix A represents the coordinates of the objects points M_i in the UAV coordinate frame. Since this study is dealing with coplanar points, the matrix A has rank 2 so additional constraints are required to form a well-posed system of linear equations. The ambiguity of solution can be seen as illustrated in Figure 4.4.

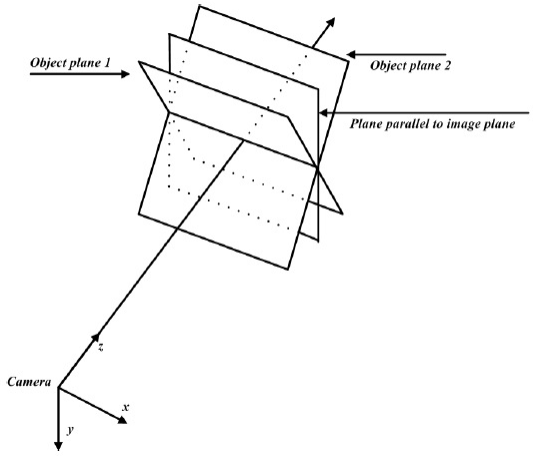
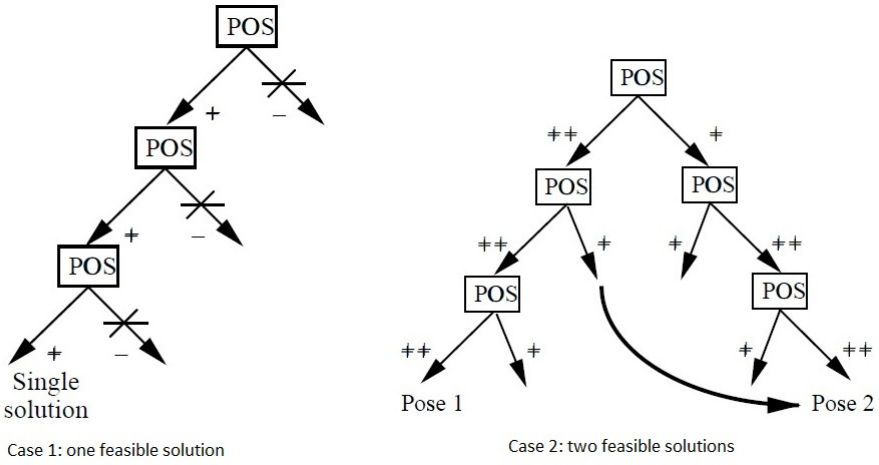


Figure 4.4: Same pose from different planes (from: [25]): In this case, the object plane can have two very different positions that returns the same $2D$ image correspondences.

When dealing with coplanar points, several poses that are very different have the same orthographic projection as shown in Figure 4.4. That is, the PosIt algorithm for coplanar points aims to find all the poses and then chooses the best match. More specifically, it finds two poses for each iteration and either picks one or continues with both options (see Figure 4.1.3). This process can be illustrated graphically as a tree with n iterations and $2n$ solutions but only two poses are kept at any time (though they may both be feasible). For each pose, it is further verified that all the points are in front of the camera (i.e., all $Z_i > 0$); otherwise the pose is discarded. More details about the method can be found in [25].



(a) (b)

Figure 4.5: PosIt Feasible Solutions (from: [25]): (a) Case 1: the algorithm yields one feasible pose represented by $+$ sign and one unfeasible pose representation by $-$ sign; (b)Case 2: two feasible solution but one of them is better then the other and they are represented by $++$ and $+$ sign, respectively.

4.2 Implementation

This section addresses the implementation of the different pose estimation algorithms explored in this thesis and their corresponding experimental results. A crucial step before using any pose estimation method is to compute the camera intrinsic parameters (a.k.a camera calibration). The purpose of camera calibration is the recovery of the principal distance parameter, i.e., the focal length, f (assuming $f = f_x = f_y$), and the principal point coordinates (c_x,c_y). To achieve this goal, an open source calibration Matlab toolbox [32] was used, where a dot pattern, such as seen in Figure 4.6, is used to compute the camera intrinsic parameters.

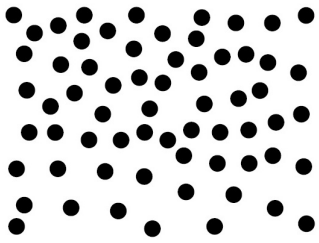


Figure 4.6: Calibration Pattern (from: [32]): The calibration toolbox uses a planar pattern comprising randomly distributed black dots where an object will be placed in different orientations to help compute the camera intrinsic parameters.

The present study uses the iPhone 4s camera with an empty plastic bottle in front of the calibration pattern in several orientations, as shown in Figure 4.2.

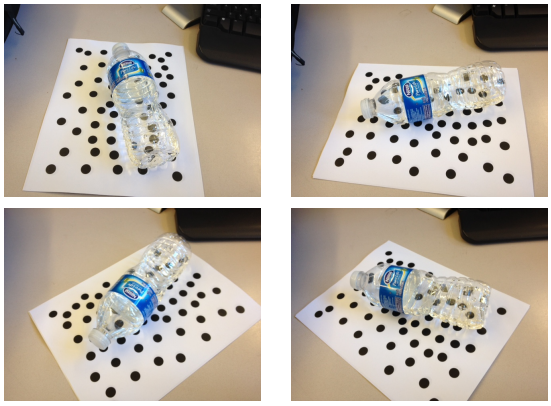


Figure 4.7: Calibration Images: The empty plastic bottle is placed over the calibration pattern and its orientation changed four times to compute the camera intrinsic parameters.

Use of the calibration matrix toolbox on the calibration images in Figure 4.2 yields the the following calibration matrix:

$$K = \begin{pmatrix} 608.7 & 0 & 314 \\ 0 & 608.7 & 320 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.36}$$

where $f = 608.7$ is the focal length and the principal point coordinates are $c_x = 314$ and $c_x = 320$.

4.3 Simulation Setup

The experiment was carried out with synthetic data for the three approaches discussed in Section 4.1. The experiment can roughly be summarized in the following steps, also graphically illustrated in Figure 4.8:

1. Generate n random points in the object plane
   - In this thesis, we use $n = 6$ reference points
2. Generate three random angles and build the corresponding rotation matrix, R
3. Generate three translations along the x, y, and z axes to form the translation vector, T
   - These three steps were performed by a Matlab function called `Generate_Data(NP)` where `NP` is the number of feature points.
4. Perform $3D$ to $2D$ perspective projections using camera calibration matrix A, the n object points, the rotation R and translation T:
   - The Matlab function `imgsynthesis(NP,points,R,T,FOCAL_L)` is used, where

points are the random points generated using the Generate_Data function described above
    - R is the rotation matrix, computed from randomly generated angles and using the Matlab function rotation
    - T is the randomly generated translation vector
    - FOCAL_L is the camera focal length from the camera calibration matrix
5. Add Gaussian noise to the image
    - This step is performed using a Matlab function called AddNoise(ImagePoints,NOISE_AMPL) where ImagePoints is a three dimensional array that contain the randomly generated features points and NOISE_AMPL is the noise amplitude
6. Use the noisy image points and the object points to estimate real rotation and translation
    - After obtaining the simulated image points and their corresponding features points, the pose estimation algorithm is used to estimate the rotation and translation vector
    - Once the rotation matrix has been found, the three rotation angles can be extracted using the function GetANG_fromRotMat(Rp) where Rp is the calculated rotation matrix
7. Compare the estimated parameters with the generated ones and compute the errors
    - The computed rotation and translation parameters are compared to the known (randomly generated) rotation angles and translation values which were used to compute the original image points.
    - The mean and standard deviation of errors between the estimated and generated rotation matrix and translation vector are computed and recorded

The aforementioned steps are repeated over 100 runs in which new random set of object points, random rotation matrix, and random translation vector are generated to evaluate the three approaches highlighted in Section 4.1. The mean error and the standard deviation for the three rotation angles as well as for the three translations are then computed. The effect of noise amplitude on the pose estimation and the execution time for each methods is also investigated. The parameter ranges used in these numerical studies are shown in Table 4.3.

4.4 Experimental Results

This section presents the results for each pose estimation method by illustrating the rotation and translation errors as well as their respective execution times.

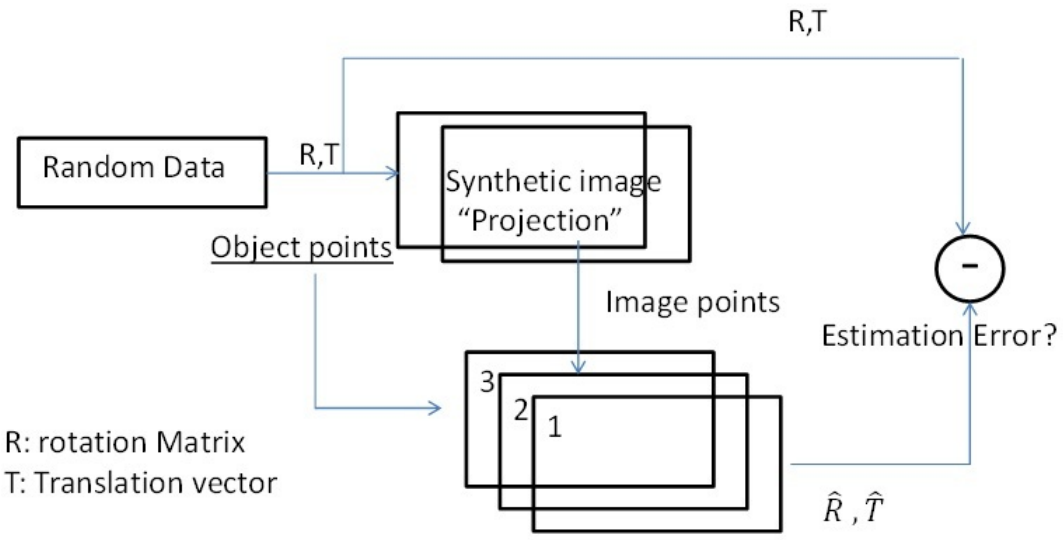


Figure 4.8: Simulation Setup: Once a rotation matrix R and a Translation vector T are generated from random data they are used to project the features points on the image plane to get the images points. Both features points and image points are fed to the pose estimation method to estimate the object pose and compare the estimated rotation and translation with the generated ones.

Parameter	Min	Max
Rotation	$-\pi/2$	$\pi/2$
Translation [X,Y,Z]	0	1000
Object coordinates		
X	-40	40
Y	-40	40
Z	0	0

Table 4.1: Parameter ranges for randomly generated data: The rotation angles are defined to be between $-\pi$ and π, the translation for X, Y, and Z to be between zero and 100. The object feature points are defined between -40 and 40 for both X and Y and zero for Z coordinates since the study is dealing with coplanar features points.

4.4.1 Non-iterative Pose Estimation Algorithm

The EPFL algorithm was executed over 100 random runs where three rotations angles, three translation values, and six features points were randomly generated. The estimated rotation angles were found to be very accurate, where their mean error (in degrees) was around zero for *roll*, pitch angle, and *yaw* angle, as reported in Figure 4.9. Concerning the translation estimation error, the EPFL non-iterative algorithm provided encouraging results, where the highest mean error was 0.390 m for the Z translation (see Figure 4.10).

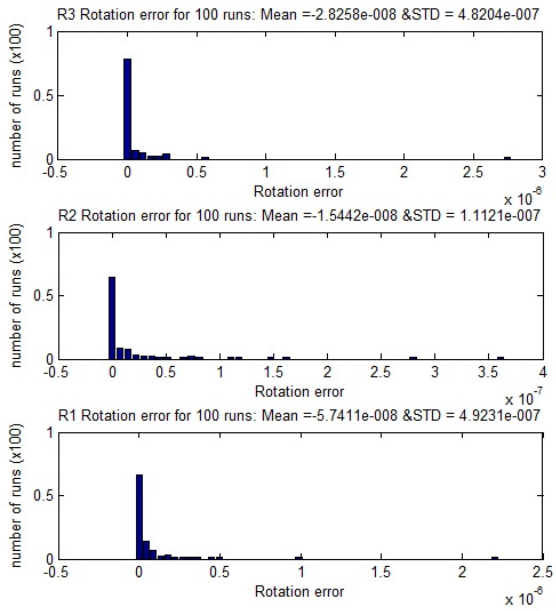


Figure 4.9: EPFL Algorithm Rotation Angles Error: This figure plots rotation errors for each rotation angle as a function of number of runs with a fixed number of feature points and a noise amplitude equal to one pixel. The mean error and standard deviation of error are computed over these 100 random runs.

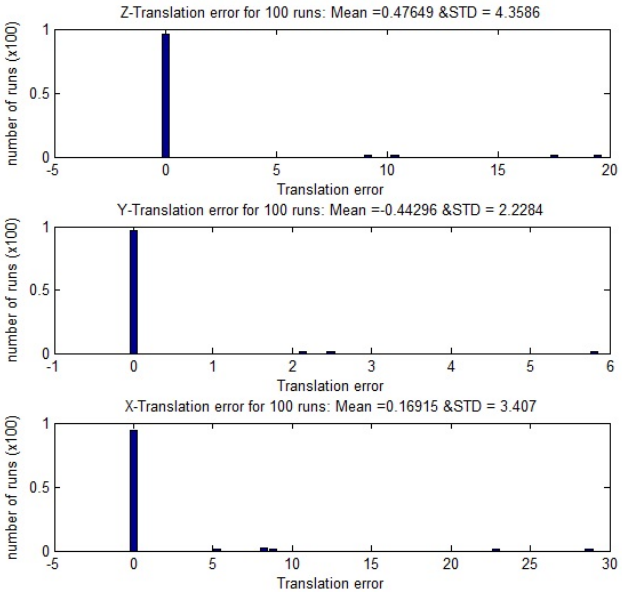


Figure 4.10: EPFL Algorithm Translations Error: This figure plots the error for the three translations along X, Y, and Z axes as a function of number of runs with a fixed number of feature points and a noise amplitude equal to one pixel. The mean error and standard deviation of error are computed over these 100 random runs

The computation cost of this method, shown in Figure 4.11, remained almost the same around 0.025 seconds for 100 random runs except for some outliers that does not exceed 0.2

seconds. (These few longer computation times may be due to the performance of the laptop computer used to perform these tests.)

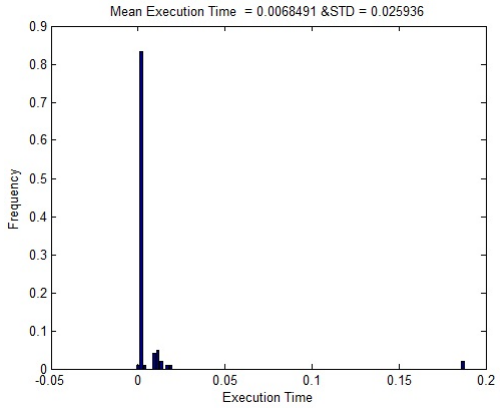


Figure 4.11: EPFL Algorithm Execution Time: This figure plots execution time (in milliseconds) for each run as well as the overall mean and standard deviation for the execution time.

4.4.2 PosIt Algorithm

This algorithm was executed over a 100 random runs where we randomly generated three rotations angles, three translations, and six features points. The estimated result for *yaw* angle was very accurate, having a mean error around zero for all 100 runs. However, the estimated result for *pitch* and *roll* angles were not accurate, having a standard deviation of 15.54 and 25.65 degrees, respectively (Figure 4.12). Concerning the translation estimation error, acceptable results were obtained for Z-translation, but not for X and Y translations. This algorithm needs further tuning to get better results to be competitive with the EPFL algorithm.

The computation cost of this method remained almost the same around 0.1 seconds for 100 random runs except for some outliers that does not exceed 0.2 seconds (Figure 4.14).

4.4.3 Homography-Based Pose Estimation Algorithm

This algorithm was executed over 100 random runs where three rotations angles, three translation, and six features points were randomly-generated. The estimated result for *yaw* angle was very accurate, with a mean error and standard deviation near zero. However, the estimated results for *pitch* and *roll* angles were not acceptable, having standard deviations of 25.48 and 28.78 degrees, respectively, as shown in Figure 4.15. The translation estimation error for Z translation was found to be acceptable, but not those for X and Y translations, which had a standard deviation of 153.52 and 66.02 meters, respectively (see Figure 4.16). The implementation of this algorithm may need further improvement and optimization to get better results as

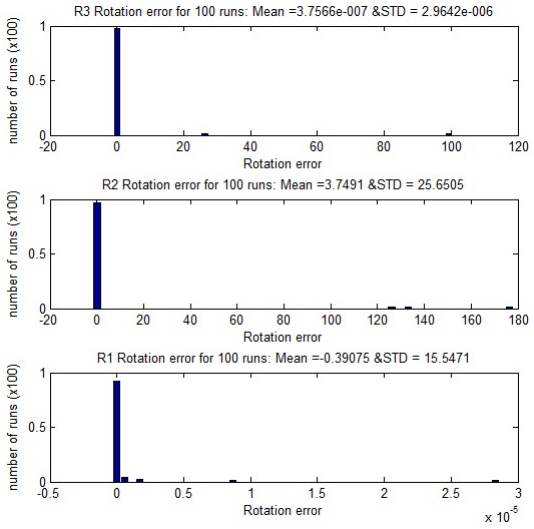


Figure 4.12: PosIt Algorithm Rotation Error: This figure plots rotation errors for each rotation angle as a function of number of runs with a fixed number of features points and a noise amplitude equal to one pixel.

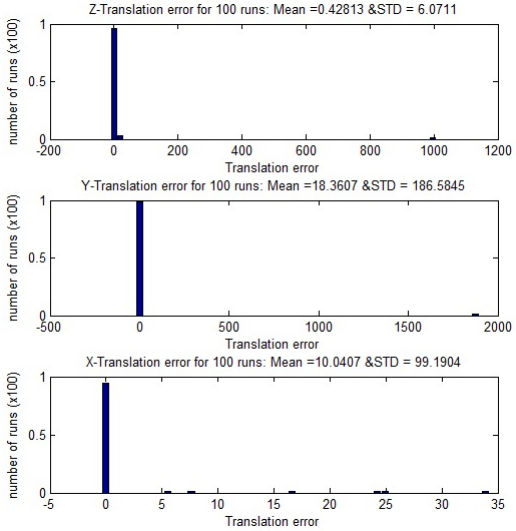


Figure 4.13: PosIt Algorithm Translation Error: This figure plots error for the three translation alon X, Y, and Z-axis as a function of number of runs with a fixed number of features points and a noise amplitude equal to one pixel.

compared to the EPFL algorithm. Finally, the computation cost of this method remained consistently around 0.15 seconds for the 100 random runs, except for some outliers that reached 0.35 seconds (Figure 4.17).

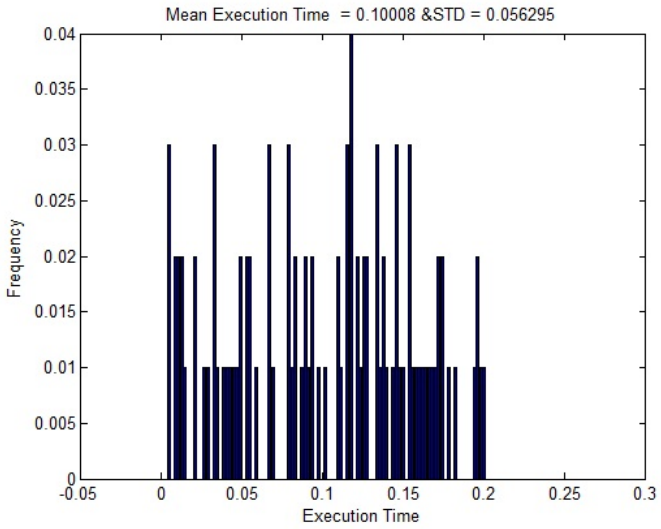


Figure 4.14: PosIt Algorithm Execution Time: This figure plots execution time for each run as well as the overall mean and standard deviation for the execution time. It was noted that the execution time for this algorithm is random.

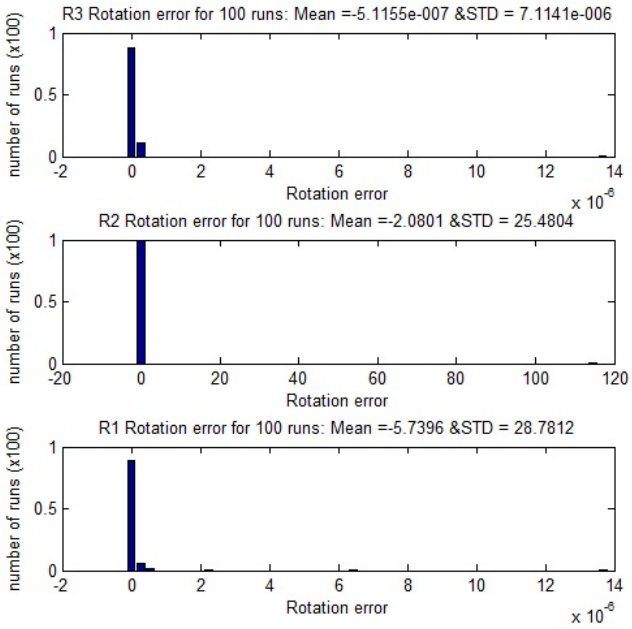


Figure 4.15: Homography Algorithm Rotation Error: This figure plots rotation errors for each rotation angle as a function of number of runs with a fixed number of features points and a noise amplitude equal to one pixel.

4.5 Summary of Results

The three pose estimation algorithms were tested over a 100 randomly generated parameters using six object features points. The EPFL algorithm showed very accurate results in estimating

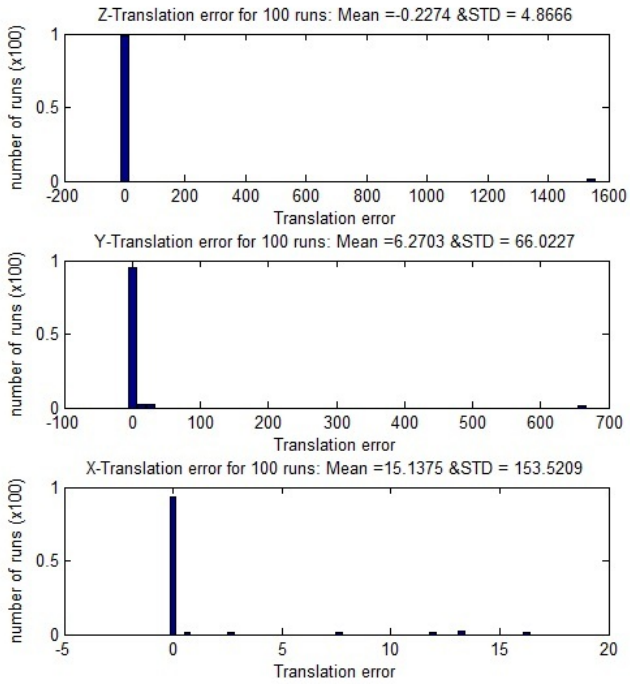


Figure 4.16: Homography Algorithm Translation Error: This figure plots error for the three translation alon X, Y, and Z-axis as a function of number of runs with a fixed number of features points and a noise amplitude equal to one pixel.

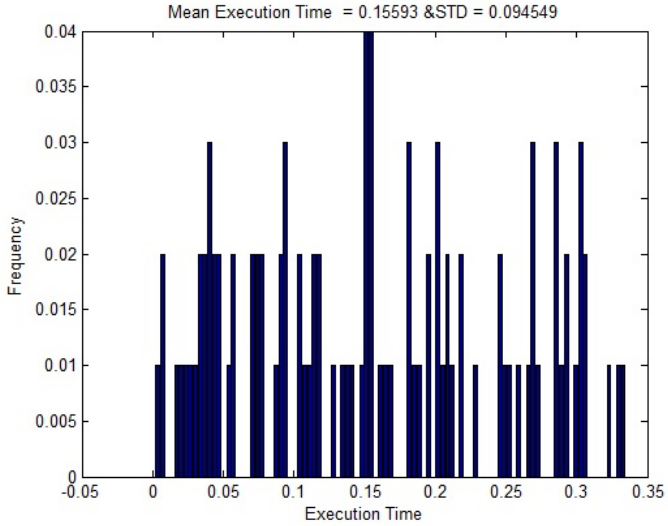


Figure 4.17: Homography Algorithm Execution Time: This figure plots execution time for each run as well as the overall mean and standard deviation for the execution time.

the pose for these coplanar points. The other two algorithms, PosIt and the homography-based approaches, showed some acceptable results on estimating Z translation and *yaw* angle values,

but failed to provide a good estimate for the remaining parameters, highlighting the need for further improvement and tuning. Table 4.2 summarizes this study's findings.

Estimated parameters	EPFL	PosIt	Homography
roll STD (degrees)	0.315	15.84	2.96×10^{-6}
pitch STD (degrees)	3.47×10^{-8}	6.30	25.65
yaw STD (degrees)	0.375	6.402×10^{-5}	15.54
X-Translation STD (meters)	3.814	59.9	151.92
Y-Translation STD (meters)	5.57	160.77	153.61
Z-Translation STD (meters)	3.92	6.69	4.11
Mean execution time (seconds)	0.0057	0.124	0.56

Table 4.2: Summary of Results: In comparing the performance of each algorithm in estimating rotation angles and translation vectors, it was noted that EPFL gives very accurate results and near real-time execution.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions and Recommendations

This chapter presents the results achieved in this thesis, the limitations and problems faced, and then highlights the eventual avenues for future research.

5.1 Summary and Conclusions

This thesis presented different computer vision algorithms for both UAV detection in still images and pose estimation of that UAV. The first task was addressed by investigating and implementing two computer vision algorithms for UAV detection. The first algorithm was based on using edge detection and image smoothing. This approach gave a high detection rate, but only works under the assumption that there is no other object in the scene having the same size as the target UAV. The second algorithm applied morphological filtering and color-based detection, where red color markers were placed on the surface of the target UAV in an arbitrary configuration. This method showed promising results but suffered from requiring constant illumination, lighting conditions, and target orientation, thereby limiting its robustness in realistic contexts.

The pose estimation task was addressed using three pose estimation algorithms. The first one used planar homography, which showed barely acceptable results; the translation standard error was around $150m$ which is not acceptable in a real UAV avoidance (or engagement) application. This inaccuracy could be due to the incorrect assumption that feature points and their image points are related via planar homography and may need better implementation and improvement. The second approach was based on a non-iterative pose estimation algorithm presented by researchers from EPFL. This algorithm gave good results for both rotation and translation (refer to Figure 4.9) and can work for both co-planar and non-coplanar feature points. The last algorithm studied represents a well-known Pose from Orthographic Projection and Scaling with Iterations (PosIt) approach; however, we find that although this algorithm gives acceptable results for the rotation angles, the translation vector achievable renders this approach less useful for real-time UAV pose estimation purposes.

5.2 Recommendations and Future Work

This thesis can be considered as an initial proof of concept and a foundation for the implementation of computer vision algorithms for both UAV detection and pose estimation. There are

several issues to be addressed and future research avenues to be considered. Matlab provided a simulation and a test environment in the lab environment. That is, an alternative implementation would be using OpenCV (open computer vision) for real-time implementation and faster computation since OpenCV has precompiled libraries that may work better for real image data.

Another avenue for future research would be to design and build light-based beacons (e.g., LEDs) mounted to the UAV body and test the detection and pose estimation in field experiment conditions. In addition, it may be possible to investigate and test different detection approaches such as the use of RANdom SAmple Consensus (RANSAC) algorithms [41], implementation of Scale-Invariant Feature Transform (SIFT) features [42], and optical flow approaches that have been shown to also have promising results for such applications.

Appendix: Selected Matlab Source Code

All source code developed and used for this thesis can be found online at `https://faculty.nps.edu/thchung` under the *Software* section.

Frame Grabbing Function

This function converts a MPEG-2 video to a sequence of frame in JPEG format at a rate of 30 frame per seconds (FPS).

```
function framegrabber(vidrostream)
%movie2frames generate frames from the given video
% vid is the video file name
% USAGE :>> framegrabber('video.avi')
%mmreader with the read method used to read video data from a multimedia
%file in our case 'avi' file
    readerobj = mmreader(vidrostream);
    vidFrames = read(readerobj);
% get the number of frames using FPS = 30 (FPS frame per seconds
    numFrames = get(readerobj, 'numberOfFrames');
% initialisation of wait bar as a debugging tool
    h = waitbar(0,'Grabbing Frames.....');
for k = 1 : numFrames
    % get the individual frames from the frame sequences
    mov(k).cdata = vidFrames(:,:,:,k);
    mov(k).colormap = [];
    %imshow(mov(k).cdata);
    imagename=strcat(int2str(k), '.jpeg');
   cd('.\frames\');
   % write each frame to the 'frames ' folder
imwrite(mov(k).cdata, strcat('frame',imagename));
    waitbar(k/numFrames,h);
fprintf(1, 'saving frame  %d:\r', k);

%extractComponents(mov(k).cdata);
cd ..
end
fprintf(1, 'Done .. : number of frames  %d\r', numFrames);
end
```

Image Cropping Function

This function extract a Region Of Interest (ROI) defined by the upper right corner of the frame where both x and y are positive.

```
%#########################################################################
% cropping : explore the frame folder and crop down them to reduce their
% size and keep the pixels that represent the UAV and get read of the
% outlier to help the detection process
%#########################################################################
% go to the frame folder and list the .png files
cd('.\images\');
files = dir('*.png');
% sort these images and count them
files = sort({files.name});
nframes = size(files,2);
    h = waitbar(0,'Cropping Frames.....');
% loop over the number of frames and resize them
for k = 1:nframes
imagename=strcat(int2str(k), '.png');
imgn = strcat('frame',imagename);
rgb = imread(imgn);
% change directory to copped folder where we will save the resized images
cd('.\cropped\');
  rgb = rgb(1200:1400,800:1800,:);
   imwrite(rgb,imgn);
    waitbar(k/nframes,h);

cd ..
end
close(h);
cd ..
fprintf(1, 'Done cropping ...\r');
```

Generate_Data function

This function generates random planar feature points, random rotation angles, and random translation vector to be used for image synthesis and to test the pose estimation algorithm.

```
function [points,Rotang,Trans] =Generate_Data(NP)
%++++ Function created by Hajri Riadh
%++++ This function generates random planar feature points, random rotation
%++++ angles, and random translation vector
%++++ Input:++++++++++++++++++++++++++++++++++++++++++++
%++++ NP: number of points
%++++ Output:++++++++++++++++++++++++++++++++++++++++++++
%++++ points: matrix of randomly generated points
%++++ Rotang: Three rotation angles
%++++ Trans: Translation vector
%### we consider here a 80x80 square (unit: cm)
X_min =-40;
X_max =40;
Y_min =-40;
```

```
Y_max =40;
for i=1:NP
points(i,:) = [ (X_max-X_min)*rand(1,1)+X_min ,(Y_max-Y_min)*rand(1,1)+Y_min, 0];
end
ang_lo = -pi/2;   ang_hi = pi/2;        % define angle interval
Rotang =[(ang_hi-ang_lo)*rand(3,1) + ang_lo]*180/pi;
trans_lo = 0;  trans_hi = 1000;         % interval for translation
Trans = (trans_hi-trans_lo)*rand(3,1) + trans_lo;
```

Image Synthesis Function

This function performs a perspective projection of the n feature points to get the corresponding image points using the randomly generated rotation matrix and translation vector.

```
function [ImagePoints] = imgsynthesis(Npoint,WorldPoints,rotationM,trans,Flength)
%++++ function created by Hajri Riadh
%++++ This function perform a perspective projection of the feature points
%++++ to get the corresponding image points
%++++ Input: ++++++++++++++++++++++++++++++++++++++++++++++++
%++++ Npoint: number of features points. In our Npoint = 6
%++++ WorldPoints: a matrix of feature point coordinates
%++++ rotationM : the randomly generated rotation matrix
%++++ trans: the randomly generated translation vector
%++++ Flength: camera focal length
%++++ Outout: ++++++++++++++++++++++++++++++++++++++++++++++
%++++ ImagePoints: a mtrix containing the resulting image points

%+++  add the translation for x,y, and z coordiantes
moved = zeros(Npoint,3);
moved(:,1)=moved(:,1)+trans(1);
moved(:,2)=moved(:,2)+trans(2);
moved(:,3)=moved(:,3)+trans(3);

%+++ perform the perspective projection
for i=1:Npoint
    for j=1:3
        for k=1:3
        moved(i,j)= moved(i,j)+rotationM(j,k)*WorldPoints(i,k);
        end

    end
end

for i=1:Npoint
    for j=1:2
       ImagePoints(i,j)= Flength*moved(i,j)/moved(i,3);
    end
end
```

```
end
```

Rotation Angle Extraction from Rotation Matrix

This function was created based on the paper of Gregory G. Slabaugh it computes the euler angles from the give rotation matrix.

```
function [rotUC,rotWC,rotZC] = GetANG_fromRotMat(Rm)
%++++ created by Hajri Riadh
%++++ inspired from paper of Gregory G. Slabaugh :"Computing Euler angles
%++++ from a rotation matrix"
%++++ url:http://gregslabaugh.name/publications/euler.pdf
%++++ this function has as imput the rotation matrix and it outpiut are the
%++++ three rotation angles

rotUC = atan(Rp(3,2)/Rp(3,3));
rotWC = atan(-(Rp(3,1)*sin(rotUC ))/Rp(3,2));
rotZC = atan(Rp(2,1)/Rp(1,1));

rotUC = rotUC*180/pi;
rotWC = rotWC*180/pi;
rotZC = rotZC*180/pi;
%+++ End of function
```

Color Detection Function

This function performs RED color detection by extracting color threshold, then applying morphological closing operation and image mask to isolate and localize the red markers.

```

function ColorDetect()
%++++++++++++++++++++ RED markers Detection ++++++++++++++++++++++++++++
%++++ This function was adapted by Hajri Riadh from the tutorial in
%++++ http://www.mathworks.com/matlabcentral/fileexchange
%++++ /28512-simple-color-detection-by-hue done by Image Analyst
%++++ Copyright (c) 2010, Image Analyst All rights reserved
%++++ This function perform RED color detection by extracting color threshold, then apply
%++++ morphological closing and image maske to keep the red band and desired
%++++ object pixel size
%++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
clc;    % Clear command window.
clear; % Delete all variables.
close all; % Close all figure windows except those created by imtool.
% imtool close all; % Close all figure windows created by imtool.
workspace; % Make sure the workspace panel is showing.
```

```
close all;
fontSize = 16;
figure;
% Maximize the figure.
set(gcf, 'Position', get(0, 'ScreenSize'));

% Change the current folder to the folder of this m-file.
% (The line of code below is from Brett Shoelson of The Mathworks.)
if(~isdeployed)
    cd(fileparts(which(mfilename)));
end


    % Change default directory to the one containing the default image
    % folder
    originalFolder = pwd;
    folder = 'C:\Users\dragonisto\studieslast\computer vision\dotm Files\UAV';
    if ~exist(folder, 'dir')
        folder = pwd;
    end
    cd(folder);
    % Browse for the image file.
    [baseFileName, folder] = uigetfile('*.*', 'Specify an image file');
    fullImageFileName = fullfile(folder, baseFileName);
    % Set current folder back to the original one.
    cd(originalFolder);


%end
% Read in image into an array.
[rgbImage storedColorMap] = imread(fullImageFileName);
[rows columns numberOfColorBands] = size(rgbImage);
% If it's monochrome (indexed), convert it to color.
% Check to see if it's an 8-bit image needed later for scaling).
if strcmpi(class(rgbImage), 'uint8')
    % Flag for 256 gray levels.
    eightBit = true;
else
    eightBit = false;
end
if numberOfColorBands == 1
    if isempty(storedColorMap)
        % Just a simple gray level image, not indexed with a stored color map.
        % Create a 3D true color image where we copy the monochrome image into all 3 (R, ↩
            G, & B) color planes.
        rgbImage = cat(3, rgbImage, rgbImage, rgbImage);
    else
        % It's an indexed image.
```

```
            rgbImage = ind2rgb(rgbImage, storedColorMap);
            % ind2rgb() will convert it to double and normalize it to the range 0-1.
            % Convert back to uint8 in the range 0-255, if needed.
            if eightBit
                rgbImage = uint8(255 * rgbImage);
            end
        end
    end



    % Extract out the color bands from the original image
    % into 3 separate 2D arrays, one for each color component.
    redBand = rgbImage(:, :, 1);
    greenBand = rgbImage(:, :, 2);
    blueBand = rgbImage(:, :, 3);

    [countsR, grayLevelsR] = imhist(redBand);
    maxGLValueR = find(countsR > 0, 1, 'last');
    maxCountR = max(countsR);

    [countsG, grayLevelsG] = imhist(greenBand);
    maxGLValueG = find(countsG > 0, 1, 'last');
    maxCountG = max(countsG);

    [countsB, grayLevelsB] = imhist(blueBand);
    maxGLValueB = find(countsB > 0, 1, 'last');
    maxCountB = max(countsB);

    % Determine the max gray level for the three bands
    maxGrayLevel = max([maxGLValueR, maxGLValueG, maxGLValueB]);

    % define threshold for each color band, our focus is on red color
        redThresholdLow = graythresh(redBand);
        redThresholdHigh = 255;
        greenThresholdLow = 0;
        greenThresholdHigh = graythresh(greenBand);
        blueThresholdLow = 0;
        blueThresholdHigh = graythresh(blueBand);
        % if it is 8 bit image we perform a suitable conversion
        if eightBit
            redThresholdLow = uint8(redThresholdLow * 255);
            greenThresholdHigh = uint8(greenThresholdHigh * 255);
            blueThresholdHigh = uint8(blueThresholdHigh * 255);
        end


    % Now apply each color band's particular thresholds to the color band
    redMask = (redBand >= redThresholdLow) & (redBand <= redThresholdHigh);
    greenMask = (greenBand >= greenThresholdLow) & (greenBand <= greenThresholdHigh);
    blueMask = (blueBand >= blueThresholdLow) & (blueBand <= blueThresholdHigh);
```

```
    redObjectsMask = uint8(redMask & greenMask & blueMask);

% define the smallest acceptable area  should be adjusted with the light
% beacon size on the image
    smallestAcceptableArea = 10;
% removes  all connected components that have fewer than 'smallestAcceptableArea' pixels
    redObjectsMask = uint8(bwareaopen(redObjectsMask, smallestAcceptableArea));

% define a morphological structuring element having a disk shape with a
% radius = 2 pixels
    structuringElement = strel('disk', 2);
    %perform morphological close operation that enlarge the boundaries of foreground (bright)↩
         regions in an image (and
    %shrink background color holes in such regions)
    redObjectsMask = imclose(redObjectsMask, structuringElement);

% fill background holes created by the preceding operations
    redObjectsMask = uint8(imfill(redObjectsMask, 'holes'));
    redObjectsMask = cast(redObjectsMask, class(redBand));
    % prepare mask for each color band
    maskedImageR = redObjectsMask .* redBand;
    maskedImageG = redObjectsMask .* greenBand;
    maskedImageB = redObjectsMask .* blueBand;
%concatenate the three color band's mask
    maskedRGBImage = cat(3, maskedImageR, maskedImageG, maskedImageB);

    imshow(maskedRGBImage);
    fontSize = 13;
    caption = sprintf('Red Beacom only');
    title(caption, 'FontSize', fontSize);
    % save the resulting image to the disk for further processing
    Img = maskedRGBImage;
    tmp = rgb2gray(Img);
    imwrite(tmp,'tempimage.gif');

return;
%++++++++++++++++++ End of ColorDetect function ++++++++++++++++++++++++++
```

Connected Components Extraction Function

The purpose of this function is to isolate connected component in the input image, extract and then locate the centroid of each connected component.

```
function conectedcomponent(image)
%++++++++++++++++++ Connected component extraction ++++++++++++++++++++++++
% the purpose of this function is to isolate connected component in the input image, extract
% and then locate the centroid of each connected  component .
I = imread(image); % read the input image
```

```
gray = medfilt2(I,[9 10]); % perform a median filter to the image to get ride of noise and ↩
    undesired effects
level = graythresh(gray); % Image thresholding using automatic threshold
bw =  gray> 0.2;
imwrite(bw,'bw.jpg'); % save the binary image in a temporary image

[L,num] = bwlabel(bw);%bwlabel returns a matrix L, of the same size as BW, containing labels ↩
    for the
%connected objects in BW.

M = im2uint8(L/num);
imwrite(M,jet,'label.jpg');% save the labeled image into label.jpg

for i =1:num

    area(i) = bwarea(L==i); %  estimates the area of the objects in binary image

end
% keep object with area >210
x = find(area>210);
s = regionprops(L,'PixelIdxList', 'PixelList','Centroid');
centroids =[s.Centroid];


% show the isoled connected component as well as their centeroid
imshow(L)
title('Red Blobs center of mass', 'FontSize', 13);
hold on
for k = 1:num
    idx = s(k).PixelIdxList;
    pixel_values = double(L(idx));
    sum_pixel_values = sum(pixel_values);
    x = s(k).PixelList(:, 1);
    y = s(k).PixelList(:, 2);
    % display the centroid of each connected component with each index
    % "Point x"
    xbar = sum(x .* pixel_values) / sum_pixel_values;
    ybar = sum(y .* pixel_values) / sum_pixel_values;

    plot(xbar, ybar, '*')
    text(xbar+2, ybar, 'Point')
end
hold off
```

REFERENCES

[1] Kathryn's Report. Cirrus SR22 mid-air collision Cessna 172 NTSB Report , 2008. http://thekathrynreport.wordpress.com/2009/08/21/cirrus-sr22-mid-air-collision-cessna-172-ntsb-report/.

[2] C. Lyle, G. Christopher, and S. Sanjiv. Avoiding Collisions Between Aircraft: State of the Art and Requirements for UAVs operating in Civilian Airspace. 2008. http://www.frc.ri.cmu.edu/projects/senseavoid/Images/CMU-RI-TR-08-03.pdf.

[3] M.D. Nelson and Gregory M. Prospective Unmanned Aerial Vehicle Operations in the Future National Airspace System. 2004. http://wwwsrv2.mitre.org/work/tech_papers/tech_papers_04/04_0936/04_0936.pdf.

[4] UAS Vision - daily news related to Unmanned Aircraft Systems and Unmanned Aerial Systems - Part 117, 2012. http://www.uasvision.com/page/117/.

[5] John Lai B.E. *A hidden markov model and relative entropy rate approach to vision-based dim target detection for UAV sense-and-avoid*. Phd., School of Engineering Systems of the Queensland University of Technology, 2010.

[6] Evan Ackerman. Global Hawk UAV to Peek Inside Damaged Reactors : Discovery News. 2011. http://news.discovery.com/tech/global-hawk-uav-damaged-reactors-110318.html.

[7] Bill. Drafts. Acoustic wave technology sensors. *Microwave Theory and Techniques, IEEE Transactions on*, 49(4):795 –802, apr 2001. ISSN 0018-9480.

[8] K. Kuchar James and Ann C. Drumm. The Traffic Alert and Collision Avoidance System. 16 (Lincoln Laboratory Massachusetts Institute of Technology), 2007.

[9] MIL Newsletter. Eurocontrol- European Organisation for the Safety of Air Navigation. 2005.

[10] Federal Aviation Administration (FAA). Automatic Dependent Surveillance-Broadcast (ADS-B). In *Federal Aviation Administration (FAA)*, 2007.

[11] D. Shim, H. Chung, H.J. Kim and S. Sastry. Autonomous Exploration in Unknown Urban Environments for Unmanned Aerial Vehicles. In *AIAA Guidance, Navigation, Control Conf. Exhibit*. AIAA, San Francisco,CA, 2005.

[12] S. Grzonka, G. Grisetti, and W. Burgard. Towards a Navigation System for Autonomous Indoor Flying. In *Robotics and Automation, 2009. ICRA '09. IEEE International Conference on*, pp. 2878 –2883, may 2009. ISSN 1050-4729.

[13] G.C.H.E. de Croon, C. De Wagter, B.D.W. Remes, and R. Ruijsink. Sky Segmentation Approach to Obstacle Avoidance. In *Aerospace Conference, 2011 IEEE*, pp. 1 –16, March 2011. ISSN 1095-323X.

[14] A. Symington, S. Waharte, S. Julier, and N. Trigoni. Probabilistic Target Detection by Camera-equipped UAVs. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 4076 –4081, May 2010. ISSN 1050-4729.

[15] Hui Guo, Chengqi Cheng, and Yubo Yang. An Automated Registration of Remote Sensing Images Based on SURF and Piecewise Linear Transformation. In *Environmental Science and Information Application Technology (ESIAT), 2010 International Conference on*, volume 3, pp. 133 –136, July 2010.

[16] Dae yeon Won and Min-Jea Tank. Light Source Target Design for Vision-based Blended Wing Body UAV Recovery. In *SICE Annual Conference, 2008*, pp. 2612 –2615, Aug. 2008.

[17] L. Mejias, S. McNamara, J. Lai, and J. Ford. Vision-based Detection and Tracking of Aerial Targets for UAV Collision Avoidance. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, pp. 87 –92, Oct. 2010. ISSN 2153-0858.

[18] D. Debadeepta, G. Christopher, S. Sanjiv, and Matthew D. A Cascaded Method to Detect Aircraft in Video Imagery. In *The International Journal of Robotics Research October 2011*, volume 30, pp. 1527–1540, May 2011.

[19] R. Carnie, R. Walker, and P. Corke. Image Processing Algorithms for UAV "sense and avoid". In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pp. 2848 –2853, May 2006. ISSN 1050-4729.

[20] S.D. Deshpande, H. Venkateswarlu, and P. Chan. Max-mean and max-median filters for detection of small targets. In *Signal and Data Processing of Small Targets*, pp. 74 –83, 1999.

[21] M. Diani, G. Corsini, and A. Baldacci. Space-time processing for the detection of airborne targets in ir image sequences. *Vision, Image and Signal Processing, IEE Proceedings* -, 148 (3):151 –157, Jun 2001. ISSN 1350-245X.

[22] R. Kasturi L. Coraor O. Camps M. Yang, T. Gandhi and J. McCandless. Real-Time Implementations of Obstacle Detection Algorithms on a Datacube MaxPCI Architecture. In *Real-Time Imaging*, volume 8, pp. 157–172, 2002.

[23] G. Welch and E. Foxlin. Motion tracking: No Silver Bullet, but a Respectable Arsenal. *Computer Graphics and Applications, IEEE*, 22(6):24 –38, Nov.-Dec. 2002. ISSN 0272-1716.

[24] E.D. Andersen and C.N. Taylor. Improving MAV Pose Estimation Using Visual Information. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pp. 3745 –3750, 29 2007-Nov. 2 2007.

[25] D. DeMenthon and L. Davis. Model-based object pose in 25 lines of code. In *International Journal of Computer Vision, 1995. IROS 2007. IEEE/RSJ International Conference on*, volume 15, pp. 123 –141, 1995.

[26] D. Oberkampf, D.F. DeMenthon, and L.S. Davis. Iterative Pose Estimation Using Coplanar Points. In *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR '93., 1993 IEEE Computer Society Conference on*, pp. 626 –627, Jun 1993. ISSN 1063-6919.

[27] Yang Yang, Qixin Cao, Charles Lo, and Zhen Zhang. Pose Estimation Based on Four Coplanar Point Correspondences. In *Fuzzy Systems and Knowledge Discovery, 2009. FSKD '09. Sixth International Conference on*, volume 5, pp. 410 –414, Aug. 2009.

[28] F. Moreno-Noguer, V. Lepetit, and P. Fua. Accurate Non-Iterative O(n) Solution to the PnP Problem. In *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1 –8, oct. 2007. ISSN 1550-5499.

[29] Leow Wee. Kheng. Camera Models. `http://www.comp.nus.edu.sg/~cs4243/lecture/camera.pdf`.

[30] Berthold K.P. Horn. Tsai's Camera Calibration Method Revisited, 2000. `http://people.csail.mit.edu/bkph/articles/Tsai\_Revisited.pdf`.

[31] Jean-Yves Bouguet. Camera Calibration Toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj/calib\_doc/`.

[32] George Vogiatzis Hernández and Carlos. camera Pose Estimation from Dot Pattern. `http://george-vogiatzis.org/calib/`.

[33] Gregory G. Slabaugh. Computing Euler Angles from a Rotation Matrix. `http://gregslabaugh.name/publications/euler.pdf`.

[34] John D. Cook. Three Algorithms for Converting Color to Grayscale. 2009. `http://www.johndcook.com/blog/2009/08/24/algorithms-convert-color-grayscale/`.

[35] Yao Wang. Image Filtering: Noise Removal, Sharpening, Deblurring. 2006. `http://eeweb.poly.edu/~yao/EE3414/image\_filtering.pdf`.

[36] K. Engel. Real-time volume graphics, 2006. `http://en.wikipedia.org/wiki/Sobel\_operator`.

[37] Mike Doughty. Graphics Color Models. `http://www.sketchpad.net/basics4.htm`.

[38] Simple Color Detection by Hue - File Exchange - MATLAB Central. `http://www.mathworks.com/matlabcentral/fileexchange/28512-simple-color-detection-by-hue`.

[39] Ryan S. Yusko. *Platform Camera Aircraft Detection for Approach Evaluation and Training*. Master, Naval Postgraduate School (NPS), 2007. 83 pp.

[40] Linda G. Shapiro Stockman and George C. *Computer Vision*. New Jersey, Prentice-H edition, 2001.

[41] D. Nister. Preemptive RANSAC for Live Structure and Motion Estimation. In *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, volume 1, pp. 199 –206, Oct. 2003.

[42] Gang Xu and Chen Ma. SIFT-NMI Algorithm for Image Matching. In *Control, Automation and Systems Engineering (CASE), 2011 International Conference on*, pp. 1 –4, July 2011.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia
2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California
3. Dr. Timothy Chung
   Naval Postgraduate School
   Monterey, California
4. Dr. Raymond Buttener
   Naval Postgraduate School
   Monterey, California
5. CAPT Jeff Kline, USN(ret)
   Chair of Warfare Innovation
   Naval Postgraduate School
   Monterey, California
6. Dan Boger, Ph.D.
   Chair, Information Sciences Department
   Naval Postgraduate School
   Monterey, California